DesignBais

Reference Guide

Version 9.1.1.3

Copyright © 2021
DesignBais International
DesignBais Pty Ltd

Chapter 1 - Introduction

Overview

DesignBais Release 8 (and above) is completely compatible with DesignBais Release 7. There is no requirement for conversion or migration of DesignBais Release 7 forms and subroutines when upgrading to DesignBais Release 8. The Responsive Design tool is included in the DesignBais toolset as of Release 8.

In this document the term Release 7/8 is used to refer to DesignBais Release 7 and above.

DesignBais is a functionally rich toolset that allows MultiValue developers to design and create enterprise wide web-based applications. There is no requirement to learn different technologies such as ASP, Java or HTML – your MultiValue expertise is all you need. It was specifically designed to be compatible with MultiValue and multi-dimensional databases.

DesignBais supports the creation of design templates. These design templates allow for creation of a standard user interface that can be easily applied throughout an application or its modules. With dropdown top menus and side bar menus as a standard feature, user navigation is simple and intuitive. DesignBais was designed to create complex and consistent enterprise applications, not just one-off forms.

DesignBais provides a zero client deployment solution. There are no HTA's, plug-ins or installs at the client side. Internet Explorer 6 or above is the only requirement for extended functionality. W3C Ajax compliance makes applications available on all compliant browsers. This means that DesignBais applications can be used anywhere anytime.

DesignBais provides corporate-grade application performance and functionality that is superior in the internet-application marketplace. Traditionally, browser-based applications perform “Submit/Accept” button form validation and only communicate with the server in this circumstance. The reason for this style of architecture is two-fold: performance limitations and the functional limitations of current browser technology.

The end result is that traditional browser-based applications delivered to the corporate desktop user have been far from adequate in performance and functionality. End-users that have become accustomed to the high performance and field-based validation of terminal-based applications have found browser-based applications to be very poor alternatives.

DesignBais has solved this issue with its unique architecture that provides end-users with performance and functionality levels that they have come to expect in the past. In fact, DesignBais significantly increases the feature set available to application developers and end-users without compromising performance.

In independent tests simulating five thousand users, DesignBais continued to provide sub-second response times against a 250 million record database on Oracle.

Service Orientated Architecture

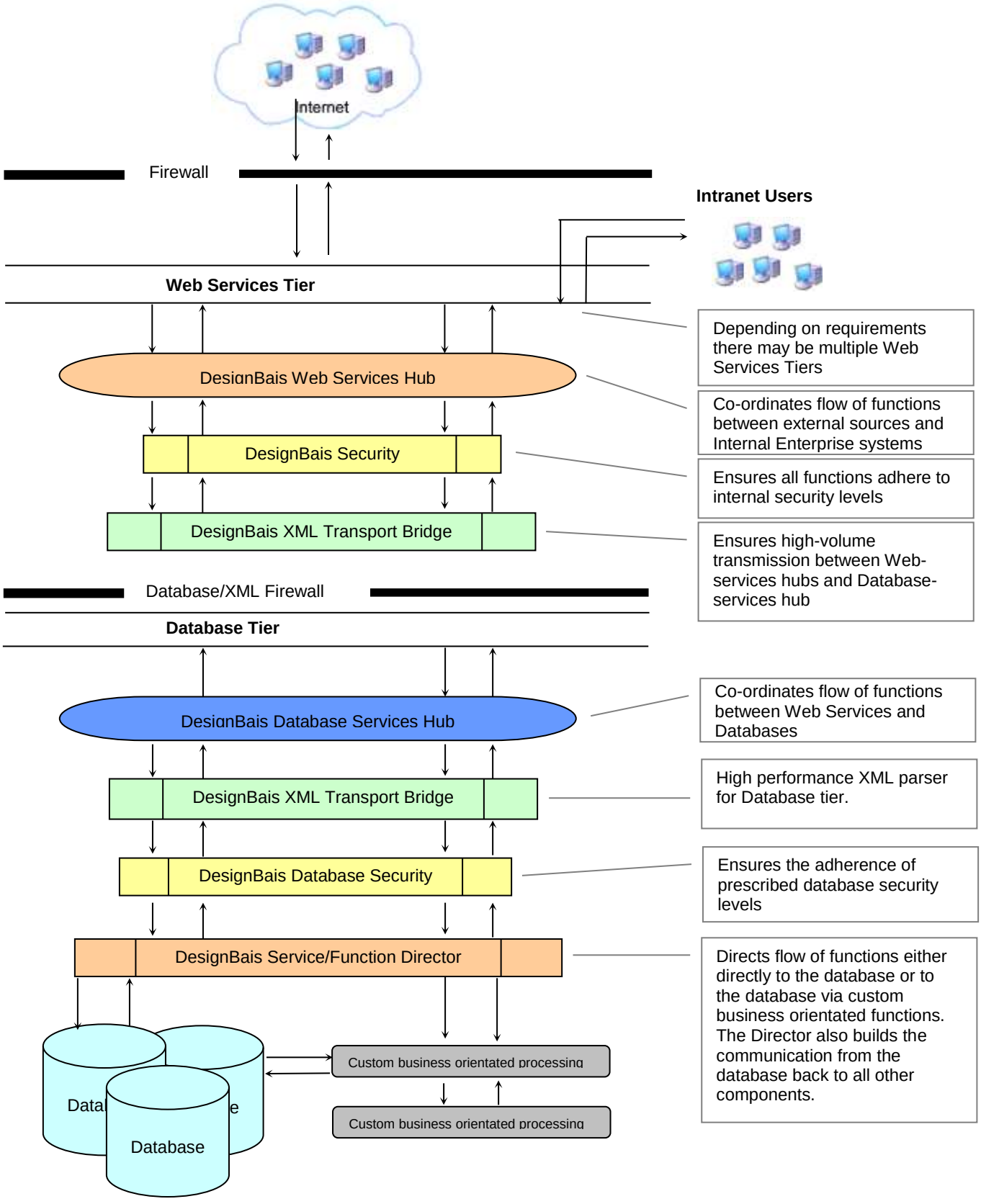
DesignBais’ architecture utilizes multi-tiered service-hubs. These hubs direct service functions to their destination for processing. Each function within DesignBais is defined within a XML transmission. Each XML transmission defines the function requested and what service hub should process the response.

DesignBais’ service orientated architecture ensures that all transmissions are sourced from a valid provider ensuring that high levels of security are maintained. With the inclusion of encrypted XML keys, DesignBais ensures that unwarranted XML is not processed by a service hub.

DesignBais' database service hub couples with the underlying database and provides native database interaction. As each XML transmission contains functional descriptors and data to support the function, the database service hub can quickly direct the underlying database to the required function. Developers can include customized routines to facilitate business rules associated with each function.

Service Orientated Architecture Schematic

The following schematic illustrates the DesignBais Service Orientated Architecture.



Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Robust Application Development Environment

DesignBais provides a sophisticated and feature rich application development environment. This environment provides functionality that significantly reduces the time it takes to develop complete applications. The DesignBais environment ensures that all applications developed meet the following requirements:

Zero deployment browser-based architecture
High performance
Commercial-grade functionality including field-based validation
Maintain the service orientated architecture
User authentication and security is maintained at all levels of the application

DesignBais was designed to significantly reduce the amount of code that developers have to generate to produce a complete application – no matter how complex. This significantly reduces the cost of application development for browser-based applications.

DesignBais has WYSIWYG development suites for both form and report generation. In addition to being very easy to use and intuitive, your application developers can quickly adopt similar techniques across both disciplines.

DesignBais enforces standards to ensure that appearance, user interface and functionality are consistent throughout an application. The use of design templates can help to ensure adherence to those standards.

DesignBais provides for complete mapping of database structures, the inclusion of business rule processing defaulted against this structure, and the splitting of validation between client and server processing. This helps to reduce server validation where simple client-based validation is all that is required. This satisfies the requirement for date, time, and numeric, range checking and mandatory validations at the client without requiring a server hit.

Normally, report production and delivery is difficult in a browser environment. DesignBais has eliminated this obstacle with its exclusive report delivery mechanisms. Reports can be previewed before printing or sent directly to a printer. With the inclusion of email distribution, reports can also be distributed via email.

DesignBais' architecture produces applications that are easy to maintain and quick to deploy. In some browser tools, you may have three levels of application source. Example: HTML, PHP, Database. This requires varied disciplines of technical staff making the development and deployment much more complicated.

With DesignBais, your existing multi-value resources are all you require. All server-side validation and server-client instruction is provided within your existing multi-value resources. With the advent of middle-ware components like OnWare, the same is true for accessing Oracle and SQL Server databases. The result is a high performance, highly functional application that can be easily migrated to any one of the major MultiValue databases as well as Oracle and SQL Server. The benefit for you is the assurance that your investment in your application is secured well into the future and not reliant on the market trends towards a particular database.

The benefit for your clients and your clients' end-users is a superior application that is extremely robust, performs at a consistently high level, is truly scalable, requires no client deployment and makes use of the functional benefits of the web browser environment.

MultiValue Support

MultiValue BASIC developers will find themselves very much at home in the DesignBais environment. You no longer need to find specialized Web developers who have no background in MultiValue applications. With a very short learning curve, BASIC developers are able to build true Web-based applications. The databases currently supported by DesignBais are:

Oracle via ONware database connector
SQL Server via ONware database connector
UniVerse (Rocket Software)
UniData (Rocket Software)
D3 (Rocket Software)
jBASE
OpenQM

The following databases may be available. Please email support@DesignBais.com

OpenInsight
Reality
Cache

DesignBais Upgrades

All DesignBais updates are distributed via the DesignBais website, www.DesignBais.com

Select the Download option from the web page and follow the instructions.

DesignBais Website Blogs

There are a series of blogs available on the Resources menu option on the DesignBais website. Many relate to the responsive design tool, some to DesignBais classic mode and some to both. The list is reproduced here to assist developers to locate helpful information by having a reference to all sources of help located in the one place.

- Hello World — a basic exercise to introduce responsive design
- Work Folders, URLs, Files and Organisation — how work folders, urls and pages are related and organized, the role of the query string parameter dbpage in targeting pages and the locations of supporting resource (res) files (css, fonts etc.)
- The Grid — Create new Rows, Columns to reflect your layout, adjust column widths, create HTML text (Element) in columns, save, publish and do the data links for your page
- Resource Files — Observe the location of res (resource) folders, create your own theme (css) file
- Adding Images — Add an image to your demo-hello page and test the responsive behaviour
- Custom Fonts — Change the font family of the the text "hello" on your hello page to "Anton". Anton is a font family that can be obtained from Google
- Page Types — Create an RD header and footer page

- Menus Adding RD menu items
- Short Cut Keys reference list of shortcut keys available in RD
- Form Samples A guide to running the RD demo forms
- Main Features at a Glance

  An introduction to working with Responsive Design
- Adding Elements to a Form

  An introduction to working with Responsive Design
- Adding a Day Picker widget to a Form

  A sample day picker demo form
- Resolving Javascript Undefined Errors

  An introduction to investigating javascript undefined errors
- Styling a Top Menu

  How to style a classic mode top menu
- Styling Dialog Boxes

  How to style classic mode dialog boxes

DesignBais Setup

The installation of the DesignBais database component will create the following database accounts:
DBINET
DBILOGIN
DBINET.DEMO
DBIHELPDESK

Refer to the separate Release Notes document for detailed instructions to load the DesignBais zip folder.
The DBISESSIONS file must have a pointer in the DBILOGIN account in order for DesignBais to function.
The DBISTATS file must exist in the same folder as the DBILICENCE file. DBISTATS is a new file in Version 7.

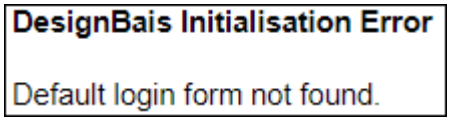
DesignBais allows users to start in any account on the system provided that the start account:
has a Q pointer to the DBISESSIONS file in the DBILOGIN account or
has a file pointer with a path that matches the path to DBILOGIN recorded in the UV.ACCOUNT file (or equivalent file on other database types).

If you wish to retain styles that originated in DBISYSFORMS on your current or a previous release of DesignBais, then it is imperative that you copy and rename the DBI account prior to loading a new release of DesignBais.

Then, after loading a new release, you may run the upgrade routine *3 Import Styles from Previous Release of DesignBais*. This is found on the *Upgrade/Migration Tools* side menu option: *DesignBais Upgrade, Upgrade Routines*. This will load, into the DBISTYLEGROUP and DBISTYLE files in the new installed release, all style groups that are in your current release and not in the new release. All styles that are used on DBIFORMS in the DBIF.FIELD.DISPLAY.CLASS.LIST that are not in the new release will be also loaded.

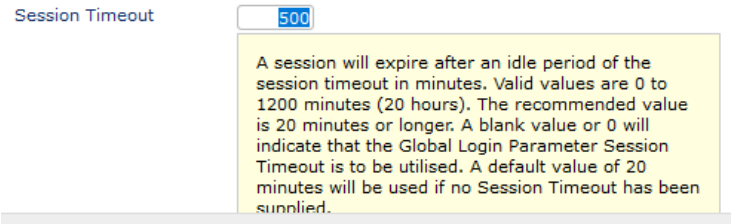
Note that if you wish to retain a standard DesignBais style that you have modified and which is stored under the same name as the standard DesignBais style then you will need to re-name it. You must assign a name that is not the name of any standard DesignBais style. This needs to be done before executing the upgrade routine.

A user record in the Users file DBIUSERS must have a start account and start form specified if a DesignBais session is to be started for this user. If there is no valid start form then the message will display:



DesignBais hit statistics have also been moved to the database and are held in the DBISTATS file.

Timeout for a user session is 20 minutes by default but can be tuned in System Parameters:



Sessions will clear eventually if they are not logged out. If a session is dropped in windows then the database has no way of knowing that the session should be released earlier than the timeout period.

The ‘requestTimeoutSeconds’ setting in the db.config file (refer to the DesignBais Web Component manual) defines the maximum time that can elapse between a request from the browser and a response from the database server. This is commonly set to a value around 30 seconds but it can be varied depending on particular requirements. See example below.

```
<entryPoint qcode="">
  <loginHost>192.168.nnn.nnn</loginHost>
  <BASUBROUTINE>BAWEBEXECNET</BASUBROUTINE>
  <loginHostType>UNIVERSE</loginHostType>
  <loginAccount>ACCOUNTNAME</loginAccount>
  <loginUser>userid</loginUser>
  <loginPassword>userpassword</loginPassword>
  <loginPublicUser>publicuserid</loginPublicUser>
  <requestTimeoutSeconds>30</requestTimeoutSeconds>
  <debugUser>debuguserid</debugUser>
  <enableXSSshield>false</enableXSSshield>
  <allowDomainNamesInLoginNames>false</allowDomainNamesInLoginNames>
  <convertLoginNamesToLowercase>true</convertLoginNamesToLowercase>
  <enableDetailedErrorMessages>true</enableDetailedErrorMessages>
</entryPoint>
```

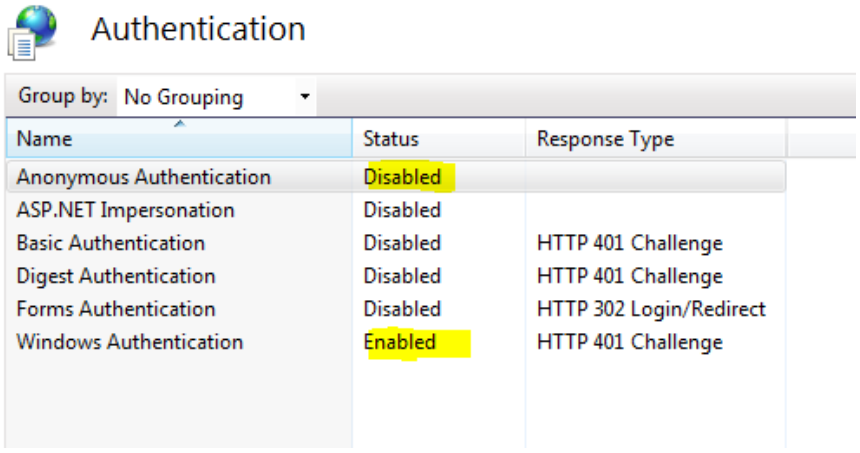
Note that the db.config file in Release 8 and above must contain a <setup> node even if there is no content. If there is no <setup> node then add:

```
<setup>
</setup>
```

to your db.config file.

Domain Name Logins

The only IIS Authentication required is Windows Authentication. Anonymous Authentication should be disabled.



The web server must be in the domain. The database server does not need to be in the domain.
The db.config file requires the following entries:

```
<allowDomainNamesInLoginNames>true</allowDomainNamesInLoginNames>
<convertLoginNamesToLowercase>true</convertLoginNamesToLowercase>
```

e.g: Domain\UserName

allowDomainNamesInLoginNames	convertLoginNamesToLowercase	result
true	true	domain\username
true	false	Domain\UserName
false	true	username
false	false	UserName

The value in the "result" column is what is provided to the application as the user name. The user Ids in DBIUSERS need to be domain\userid if convert to lower is true as shown in row 1 of the table above.

DesignBais Licencing

Licencing is controlled in the database component of DesignBais. It is tied to the database serial number. The record in DBILICENCE is called LICENCE.NET.

To apply for a licence send an email to DesignBais Support support@DesignBais.com.au. In the email you must advise the web server machine name and the Database Serial Number.

There are several methods of obtaining a licence to run DesignBais.

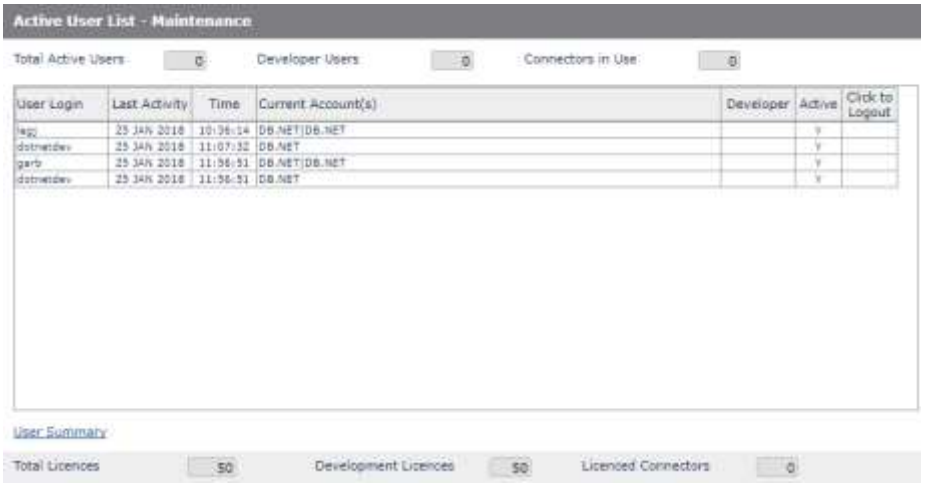
There is a User Limit type of licence which defines the number of users that can access the system concurrently. The number of Development users is also defined in this licence. As an example there could be 50 users in total with 2 development users.

A Development user is one that uses Forms or Report Designer. Such a user is included in the count of developers until they either log out or the browser times out.

A DesignBais user is a browser on a PC and this user can have multiple tabs open at the one time. Connecting with IE and Chrome from one PC will count as two users but 10 tabs in Chrome is still just one user.

Alternatively there is a Licenced Connectors licence. This uses a formula to determine the average number of concurrent users over a set period. This type of licence allows for peaks of activity where the total number of users may be much higher than normal, providing that the average of logged in users over a period is maintained.

The browser is what we use to count users. Logout or Closing a tab or Closing the browser simply removes the entry from DBISESSIONS and the active user count. This frees up a DesignBais licence. The browser record is also deleted from DBISESSIONS. The tab or SESSION.ID records remain on DBISESSIONS. This means that on the next hit from that browser the user count will be updated as the browser tab is added back into the Active User list (if the count allows) and the tab session can pick up from where it left off. The reactivated tab is added into the count or, if a new tab is opened, a new tab session is started.
The list of *Current Accounts* will not match until all tabs have had a hit and have been added back into the list.



If DesignBais were to leave the browser record and simply remove it from the Active user list when we have a logout event then this would reactivate the account list on any new hit from the browser. But the list may contain “dead” sessions (and accounts) that can never be reactivated.

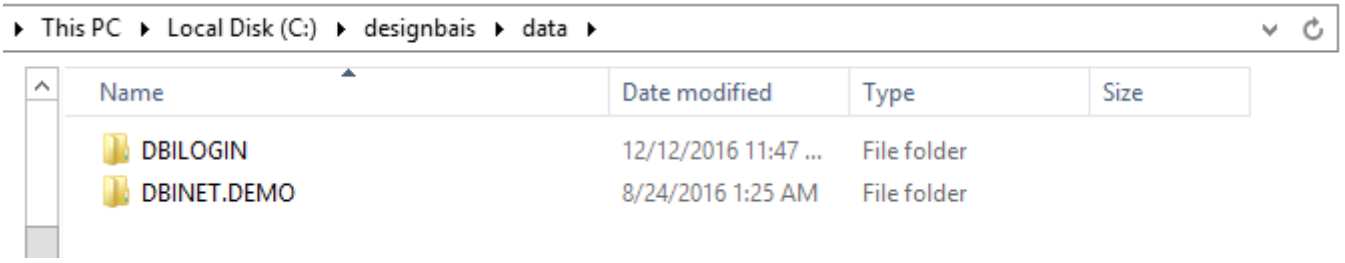
For example, if the browser is closed the session records will remain on file (until purged) but never be reactivated. It is not until DesignBais detects a hit that it knows if the session in an account is active or not.

The developer count is by browser. If the user runs any DBIFORMS_D1… or DBIREPORTS_D1… form they are added to the developer count if possible. The developer flag is held in the active user list and on the cookie. It is not cleared if the user exits a developer form because that would require DesignBais to check all active sessions to see if any others are in a developer form. This is too big an overhead.

Installing DesignBais in a Database Account

For UniVerse running on Windows:

1. Create a directory (account) in the DesignBais data directory.

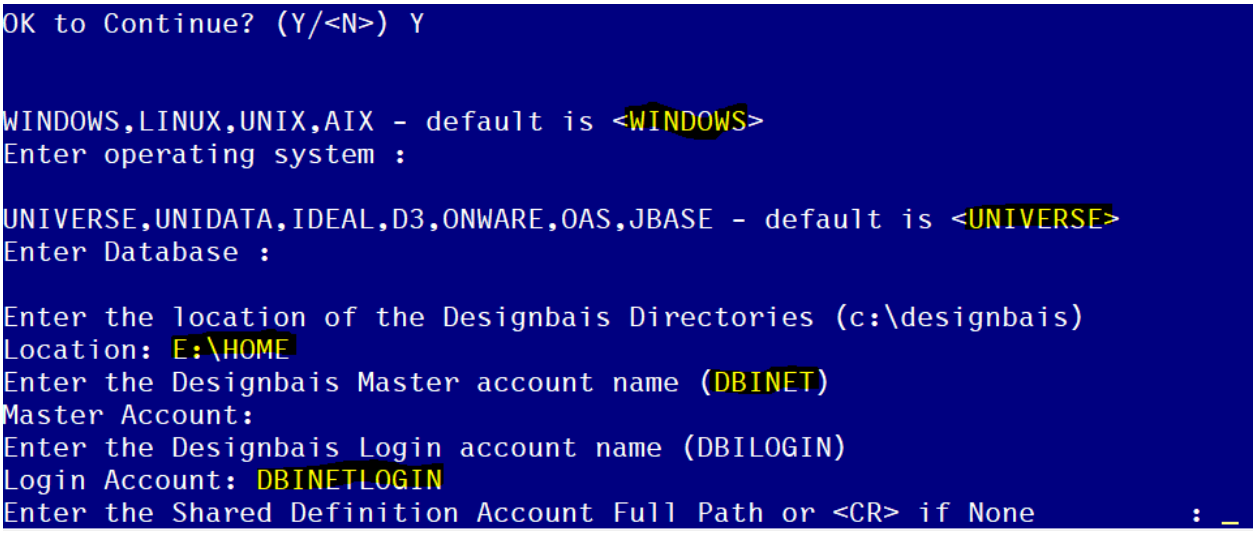


2. Telnet into the UV account and login as an Admin user.
3. UniVerse will display the following prompt:
   This directory is not set up for UniVerse.
   Would you like to set it up (Y/N)?

4. Enter 'Y' and UniVerse will prompt for:
   Ideal UniVerse compatibility
   IN2 compatibility
   Prime Information compatibility
   PICK compatibility
   PI/open compatibility
   Microdata Reality compatibility

   Which way do you wish to configure your VOC?

5. Enter "3" for PICK compatibility. This will create the files needed for the directory to be a UniVerse account.
6. Add this directory name to the UV.ACCOUNT file in the UV account by creating a record with the directory name as the key and the path to the account in attribute 11.
7. Logto DBI.
8. Compile and catalog the program DBI.P.ACCOUNT.SETUPNET.
9. Logto the new account. Catalog DBI.P.ACCOUNT.SETUPNET. This is not required if you have globally catalogued it in the previous step.
10. Run DBI.P.ACCOUNT.SETUPNET.



11. You will be prompted as shown above.

12. The location is the parent directory for the DesignBais “ba” and “data” directories. If you used the defaults when installing the database components then this directory will be “c:\DesignBais”.
13. The Shared Definition Account is used if you are going to share the DesignBais files from another account. If you enter a full path here then the account setup program will create pointers for each of the DesignBais files instead of creating the files in this account.
14. The account setup program will create files and file pointers needed to run DesignBais in this account. When it has finished you can enter this account name in the list of Start Accounts in your User Groups and / or User maintenance form. This then allows users to logto the new account.

For D3 the prompts in step 11 above are slightly different:

"Enter the DesignBais master account name (DBI) : "
"Enter the Shared Definition Account Name or <CR> if None : "

DBI.P.ACCOUNT.SETUPNET

This program has an option to create new and missing DBI files and catalog any DBINET routines that are not cataloged in the account in which this program is run. This is useful when upgrading to a new DesignBais release when there may be, say, several new DesignBais files or DesignBais subroutines.

To run this option enter “Y” or “YF” at the prompt highlighted below:

DBI.P.ACCOUNT.SETUPNET - Setup DesignBais in this account
Enter END or Q to exit at any prompt.
Operating System : WIN
Database : UNIVERSE
Use DBI.P.UPDATE.ACCOUNTNET to copy A,S,F dict items from DBINET
Defaults are stored in DBIGLOBAL or DICT,DBINET item CUSTOM.INPUT
OK to Continue? (Y/<N>) Y

Create new DBI files only (do not update existing pointers)
& only catalog missing programs (YF to only create new files) (Y/YF/<N>)

Version Notes

Important - Please read before you use Version 7

Warning

Version 7.0.0.0 will only function for 60 days if your DesignBais licence has not been upgraded to include a current maintenance certificate.

Version Notes

You must ensure that you have received a valid licence with a current maintenance certificate before installing 7.0.0.0 on a live server. If you are in doubt of the current maintenance status for one of your servers, please email support@DesignBais.com.

- In this version there is a Build Identifier in the Common Block Subroutines DBI.COMMON and DBI.SUB.COMMON

  The Build Identifier for 7.0.0.0 is **1731344905**

  You must ensure that your common block inserts have these build numbers. If not, there may be some new functionality in 7.0.0.0 that will not perform as expected.

- You will **not** have to recompile your old DesignBais source code to be compatible with version 7.

- There may be new programs in version 7.0.0.0 than were not in previous versions. For DesignBais to function properly, you will need to perform the following in all active accounts:

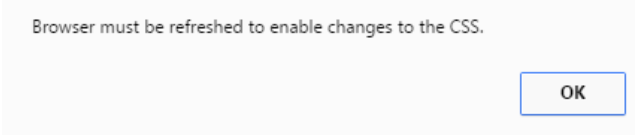
  QSELECT DBI PROGRAMS
  CATALOG DBI

  QSELECT DBINET PROGRAMS
  CATALOG DBINET

  Note that running DBI.PACCOUNT.SETUPNET will catalog the two libraries.

- The Style Group for DesignBais is now named dbaisWeb. When you first start DesignBais Release 7/8, it will have the incorrect fonts displayed. You must perform the following to correct this:

  Run Style Definition form.
  Select any style by clicking the Style search label.
  Click the **Submit** button.
  Click OK on the message advising that the browser must be refreshed.

  Refresh your browser (enter CTRL F5)

- Occasionally DesignBais forms remain in the file DBIFORMS file. This only causes a problem during upgrades. To get the new versions of all DesignBais forms, perform the following for the DBIFORMS file in each of your data accounts:

  SELECT DBIFORMS = “DBI]” “DBFINDEXDEFN]”
  If any of your application filenames start with DBI then you will need to exclude your forms from the selection
  DELETE DBIFORMS
  All forms for Release 7/8 will then be sourced from DBISYSFORMS.

- There is a new subroutine from Version 7 onwards DBI.G.RESEQ.FORMDATA. This routine must be employed when modifying forms at run time and using DBFORMLOCAL. Refer to the reference manual for full details.

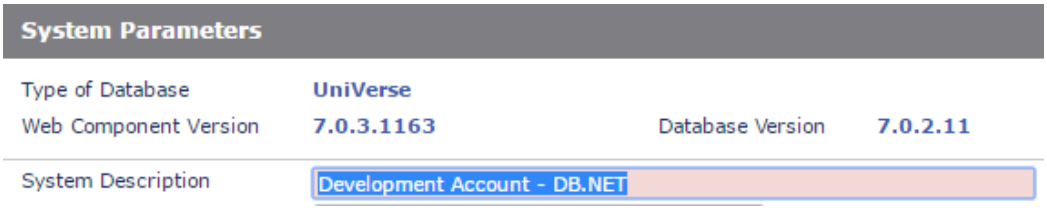
- There is a Version 7 upgrade menu that is accessed from the side menu Global Parameters, Upgrade Details, Upgrade Routines. Refer to Upgrade Details in this manual.

Web Server Component

The webserver component no longer requires installation but rather is a simple folder copy.

DesignBais Release Number

The DesignBais Release numbers are displayed on the System Parameters form.



From Version 7 onwards the release number has 4 parts.

Compatibility between Webserver and Database Server versions are based on the first two digits of the release number. This ensures that a 7.1.4.n webserver version will be compatible with, say, a 7.1.5.n database version. If either of the first two digits of the Webserver release number and the Database release number is different, DesignBais will not operate. This scenario will generate a warning email like the following example:

The browser will display the following:

Changes to DBIFORMS Fields

The changes to the field properties of DBIFORMS In Version 7 are documented here to provide developers with an indication of the potential impact on applications which utilise the ability to modify forms at run time.

Field Name	Screen Label	Attribute	Field Multivalue	Notes
DBIF.TAB.INDEX	Input Fields Use Tab Index	114	1	
DBIF.TAB.INDEX.INCR	Tab Index Increment	114	2	New field
DBIF.FIELD.DBHBMODE	Field Hit Blocker Mode	131		New field
DBIF.FIELD.HTML.ATTRIBUTES	Custom Attributes	132		New field
DBIF.FIELD.ENCODE.HTML	Encode HTML	141		New field
DBIF.OVERLAY.REQUIRED	Overlay Required	174	1	
DBIF.FORM.CENTERED	Form Centered	174	2	New option 'Inherit' (from System or Global Parameters)
DBIF.MOUSE.EVENTS	Mouse Events	174	3	
DBIF.CALC.DEPTH	Calculate Form Depth	174	4	New field

New Features in Version 7

1. Full cross browser implementation.
2. Form Section Control. Form sections are now automatically sorted so setting up section collapsing is much simpler. Section names that no longer have any fields are removed. Subsections are sorted to the end.
3. Form Tab Indexing. There is now the ability to set an increment greater than 1 between tab index settings which facilitates the insertion of new form elements within the existing sequence.
4. There is now a click timeout parameter to allow a delay for a click event to complete.
5. The Encode HTML option enhances security against XSS injection attacks.
6. Hit Blocker allows form elements to be disabled and following events to be blocked.
7. Custom Attributes for a field (added to the HTML tag and then available for javascript).
8. DesignBais styles (class) have been completely reviewed and given standard names. There are now only two Style Groups, one for forms (dbaisWeb) and one for reports (dbaisRep).
9. Ability to replace the grid control symbols (+, >, x) with your own symbols (which may be HTML).
10. There is a default button class that is applied to any button with no display class definition.
11. Menu styling improvements with the ability to control container size.
12. Phantom Status, Hit Status and Display Como options on Active Users give visibility and control of processes. Long-running reports and other phantom processes can be reviewed and killed.
13. Exclusive Locks display per account.
14. All Global Parameter functions are now maintained via forms rather than editor.
15. Form Compare routine provides developers with ability to display differences between two forms.
16. Comprehensive improvements and additions to the Reference Manual.
17. Improved File Upload:
    - allows multiple file uploads
    - is not sensitive to app pool recycling
    - is not prone to session hijacking
    - provides encoded virtual path
    - provides the option to assign unique file names

- shows only those files having allowed file extensions when picking a file using the Windows File Explorer

18. Improved CAPTCHA.
19. HTML Editor.
20. DesignBais can now provide a web service.
21. DesignBais can access an external web service such as a SOAP service.
22. New Date Picker calendar display.
23. New and much improved Code Editor.
24. Session restoration.

Data Entry Conventions

DesignBais supports standard browser techniques for data entry. There have also been a number of additions from release 4.3.3 that will make the data entry process, much faster for users by reducing the need for a mouse.

The Tab Key The Tab key is used to submit a change to a field for validation. This change event will fire a client-side validation, and/or a server-side validation event.

The Enter Key The Enter key may be used to submit the contents of a form for server-side validation. This is set-up at the form-level and is linked to a button procedure.
It may also be used to select a row number in search processes. *See KeyBoard Driven Searches.*

The Control+Enter Keys

This key combination may be used to invoke a button process that is linked to an individual field. As this feature is linked to a button, it may be used multiple times on a form.

Shortcut Date Entry

There are a number of ways to input a date into a DesignBais form at runtime.

T	Enters today's date
+*n*	Date + number or days
-n	Date – number of days

MM/DD (U.S.A) DD/MM (International)	Day + Month + Current Year
MM DD (U.S.A) DD MM (International)	Day + Month + Current Year
MM/DD/YY[YY] (U.S.A) MM/DD/YY[YY] (International)	Day + Month + Year (2 or 4 digit)

4 apr	4th April current year
4 APRIL	4th April current year
APR 4	4th April current year
April 4	4th April current year

Invalid Characters in DesignBais

The following characters are not permitted in data that is processed within DesignBais:

CHAR(0) through CHAR(8)
CHAR(11) through CHAR(12)
CHAR(14) through CHAR(31)

If these characters are present the following error message will display:

"The record that you are attempting to display contains invalid characters. Please consult your systems support. Do not attempt to update this record."

The set of invalid characters is held in the DesignBais common variable *UNPRINTABLE.STRING*.

When the above error message is displayed the offending data string is written to the sessions file with the invalid characters replaced by a null string. In other words the invalid characters are removed.

If the error is encountered by DesignBais when processing a form then the sessions file is F.DBISESSIONS and the key of the record containing the string is *TESTSTRING*.

If the error is encountered in an application subroutine that is calling DBI.G.CONVERTNET then the main sessions file is F.DBISESSIONS.MAIN and the key of the record containing the string is *BadCharacters.*

The 4 character string *"; "* (semicolon followed by 3 spaces) is used by DesignBais as a javascript command terminator. This string must be avoided in data that is entered into a DesignBais form.

HTML encoding may sometimes lead to an undefined error. For example HTML encoding of the string ***'= "M" '*** becomes '= "M" '. A double-quote followed by 3 spaces, as in this example, '= "M" ', is rendered as '= "M" ' when HTML encoded. This produces the string '; ' (semi-colon followed by 3 space characters) which is the DesignBais javascript *end of command* character.

To track down these types of errors follow this procedure:

- Clear the debug.txt record
- Run the form to the point *just* before the action that causes the error
- Delete all records from the *debug* folder on the website
- In the web form trigger the error and ensure no further browser actions occur
- Inspect the *debug.txt* file from the debug folder on the website

Note that the error message *"***WARNING*** Too many files in debug folder. Logging will stop for 5 minutes***"* means there are more than 1000 files in your */debug* folder. Logging of errors will stop until you delete some records.

IIS Response Buffering Limit

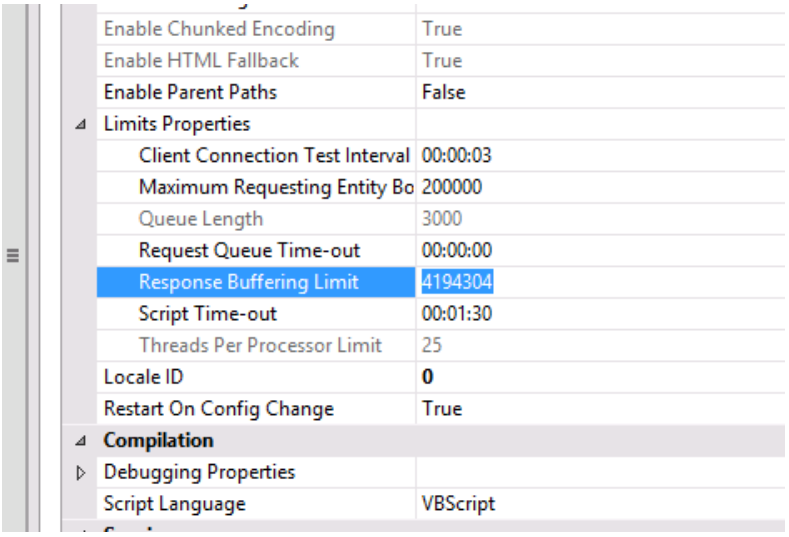
IIS has a Response Buffering Limit.

The Response Buffering Limit governs the amount of data that DesignBais can send out to the browser from the web server.

The effective limit for the number of characters that can be sent by DesignBais is 1.9 million.

WideChars occupy two bytes so, counting 1.9M characters equates to 3.8MB. Adding DesignBais overheads arrives at the limit of about 4MB.

The DesignBais data component sends large volumes of data as as discrete BLOCKs.



A malformed xml string type error may result if the volume of data exceeds the buffering limit. The following is an example of this error on a D3 system:

"errLOG_hoscaresupport__WOLPER\HoscareSupport_rad56DB9.txt 24/10/2017 4:03:33 PM errLOG_hoscaresupport_rad62755.txt"
Error description:Internal error.
Error Number:1
Error Category:406
Native error:Object required
Modal:no
Number of attempts on OK button:1
Details:ERROR:errLOG_hoscaresupport_rad62755.txt CDB0092 A malformed xml string has been received.

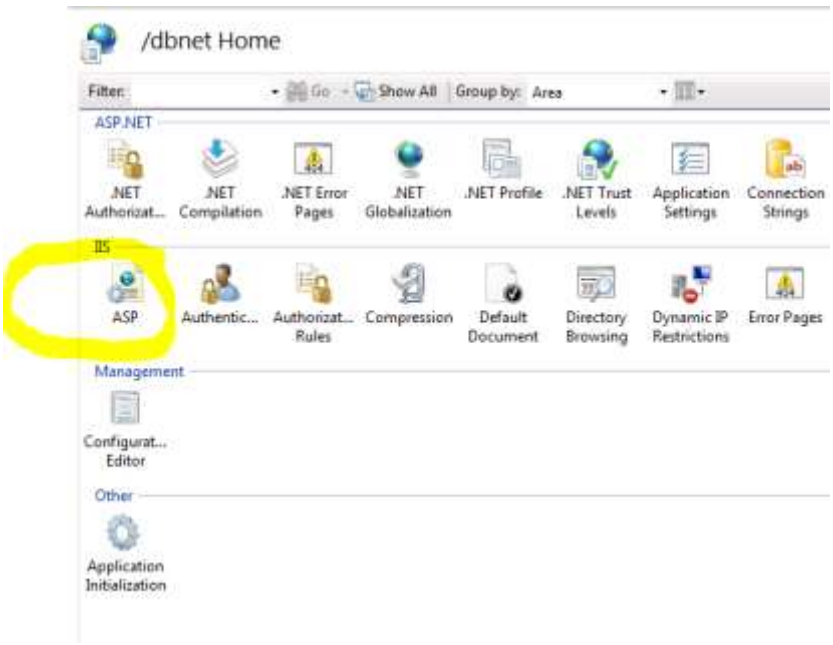
If you need to allow large file uploads/downloads to/from the server, open a CMD shell *as Administrator*, change directory to C:\inetpub\adminscripts and run the following two commands:

cscript adsutil.vbs SET w3svc/ASPMaxRequestEntityAllowed 50000000
cscript adsutil.vbs SET w3svc/aspbufferinglimit 50000000

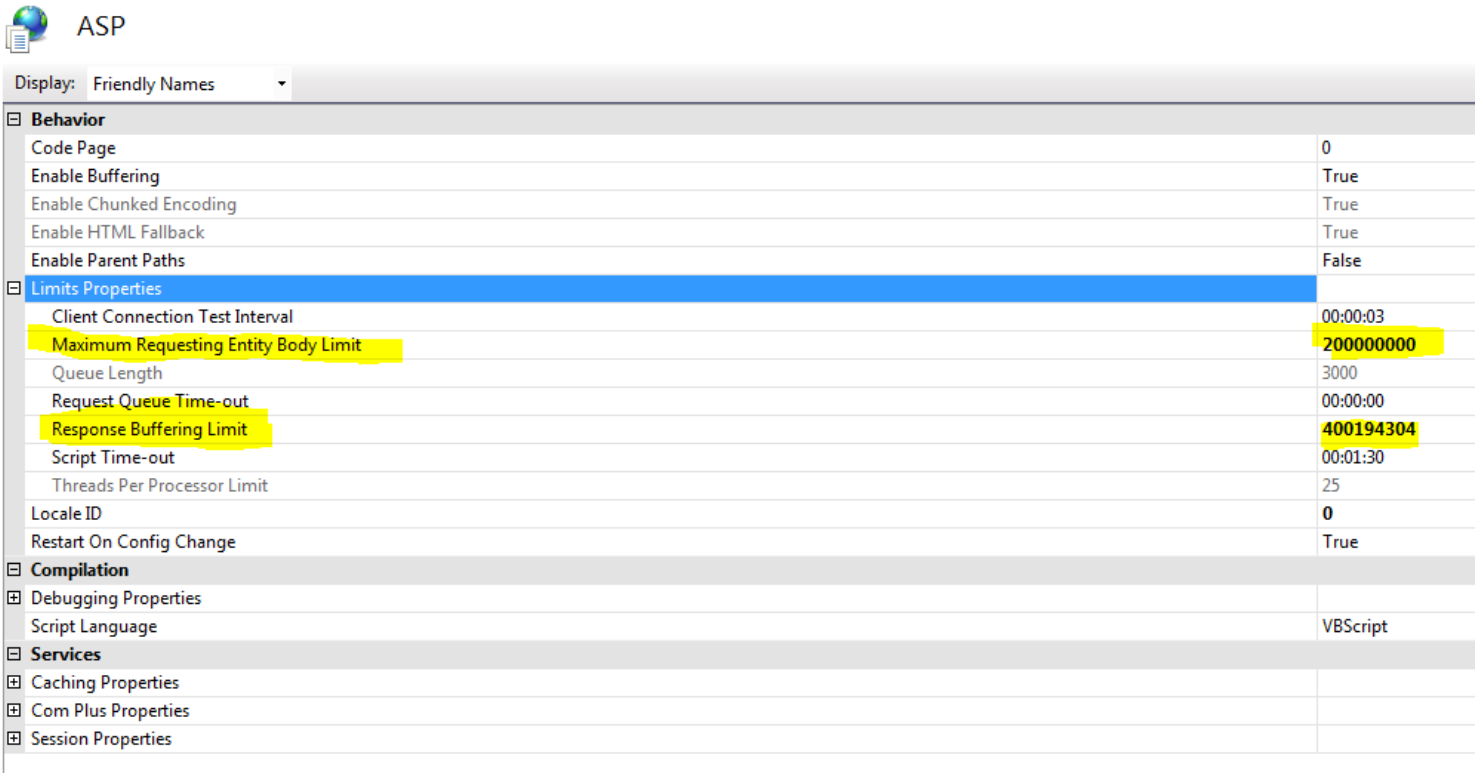
This will allow upto (approx.) 50MB file uploads.

These values can be set in the IIS Management Consiole as well.

On the left pane, select your DBNET. On the right pane click ASP as shown below.



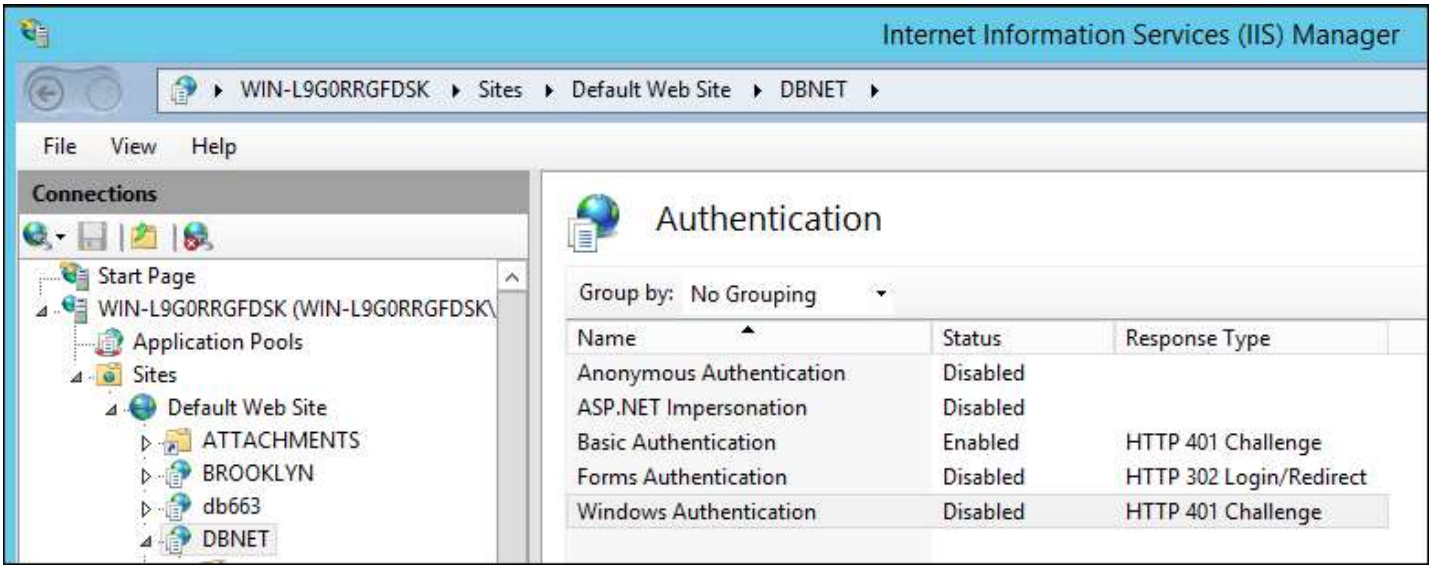
Then set the two parameters shown below to 50000000 (50MB approx).



Authentication Methods

The authentication methods in IIS that are utilised by DesignBais are:

- Basic
- Windows
- Anonymous



DesignBais reads the *db.config* file to determine the *loginAccount* of the Username supplied by the Authentication method. DesignBais will then start in the account defined in the *loginAccount* parameter in *db.config*. In the example below DesignBais will start in the account named DBINET.DEMO.

```
</setup>
 <entryPoint qcode="">
  <loginHost>172.31.29.120</loginHost>
  <loginHostType>Universe</loginHostType>
  <loginAccount>DBINET.DEMO</loginAccount>
  <loginUser>DesignBais</loginUser>
  <loginPassword>DesignBais</loginPassword>
  <loginPublicUser>DesignBais</loginPublicUser>
  <requestTimeoutSeconds>60</requestTimeoutSeconds>
  <debugUser>DesignBais</debugUser>
  <enableXSSshield>false</enableXSSshield>
  <allowDomainNamesInLoginNames>false</allowDomainNamesInLoginNames>
  <convertLoginNamesToLowercase>true</convertLoginNamesToLowercase>
  <enableDetailedErrorMessages>true</enableDetailedErrorMessages>
 </entryPoint>
```

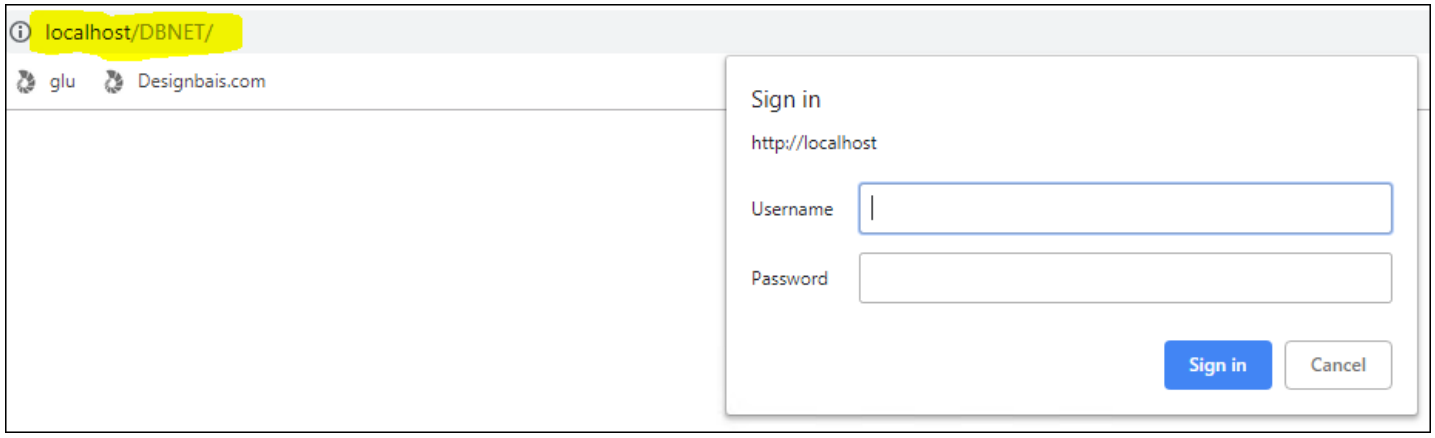
Note that the *loginUser* must NOT be a domain user. It must be a local user on both the IIS server and the database server.

Basic Authentication

Basic Authentication is used when your PC is not on the same network with the web server. In this mode, you either create local Windows users on the web server or, in IIS, configure the basic authentication against a domain. Basic Authentication must not be used without SSL. It's also for closed environments. It is not safe for public access websites. Note that you need to install the Basic authentication module first as per Section 4.4 of the Web Component manual http://www.DesignBais.com/usermanual/usermanual.aspx.

With Basic Authentication the user will be prompted for a Windows User Id and password when first accessing DesignBais. The user must be a local Windows user on the web server. If Basic authentication is set against the webserver's domain then the user must also be a part of that domain.

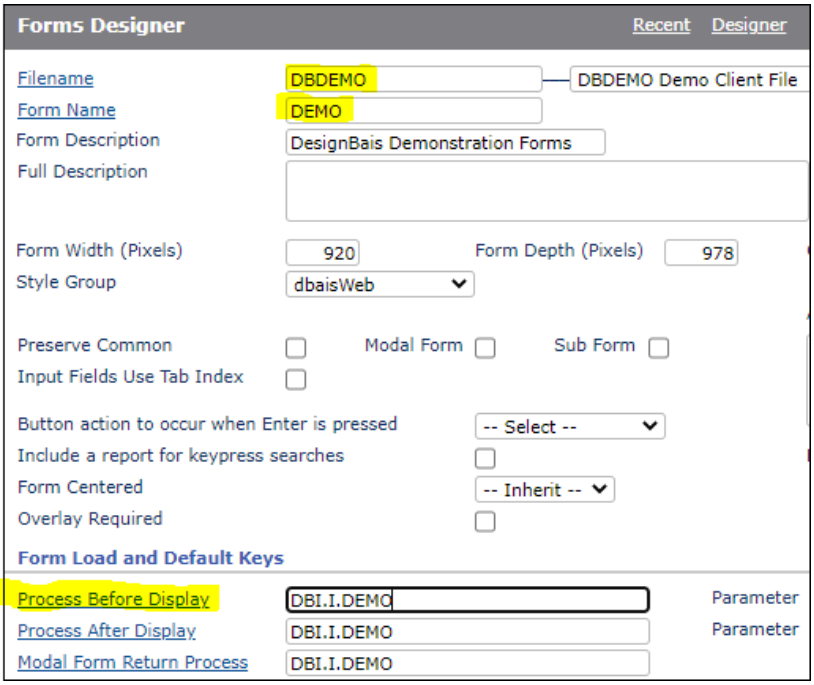
There must be a user record with the same User Id in the DesignBais DBIUSERS file in the *loginAccount*. This user record must define a start form in the account defined in the *loginAccount* parameter in *db.config*. There can be more than one start account / start form combination in the user record. DesignBais will select the first start form that it finds for the *loginAccount*, and it does not have to be the first one in the list.



In this example when entering the DesignBais url (the example is highlighted in yellow) the Sign In prompt for Username and Password is displayed. The Username entered here must exist on the DBIUSERS file in the *loginAccount*.

In this example the user will be logged into the account named DBINET.DEMO and the DBDEMO_DEMO form will display.

This start form may be set up to have a subroutine in the *Process Before Display* slot which will trigger the *BEFORE DISPLAY* event. Within this event process the developer can re-direct the user to another form. In the example below the form DBIFORMS_EXPRESS will be the first form to display.




```
157 *
158 BEFORE.DISPLAY:
159 *
160       IF WEBLOGON = 'garb' THEN
161          PROCESS.STACK = 'DBIFORMS_EXPRESS'
162       END
163       RETURN
164 *
165 AFTER.DISPLAY:
166 *
167       GOSUB DEMO.FORMS.LIST
```

Windows Authentication

Windows Integrated Authentication is used if you're on a PC that is on the same network as the web server. In that mode you don't get a login prompt because you've already logged in to the network (when you logged in to your PC with a domain). Windows Integrated Authentication is recommended for closed environments (no internet access, no public access etc.). It is not safe for public access websites.

When entering the DesignBais url with Windows Authentication in force there is no prompt for a username.

For Windows authentication, the user must be a user of the webserver's domain.

The Windows Username of the logged in Windows user must exist in the DBIUSERS file in the *loginAccount*. The procedure described above for Basic Authentication then applies.

When "Windows authention" or "Basic authentication against a domain" are used, the user login name is captured in the form of domain\username. If, in DesignBais, users are set without domain names then the db.config file must include:

<allowDomainNamesInLoginNames>false</allowDomainNamesInLoginNames>

If DesignBais DBNET is required to operate in Windows authentication mode and only in Windows authentication mode then anonymous and basic modes can be disabled.

In the web.config the authenticated user (Windows authentication) should be entered as follows:

- If the user is a domain user then domain\username syntax is required.
- If the user is a Windows user on the web server then .\username should be used (as shown below)

```
<!-- CODEEDITOR FOLDER -->
 <location path="codeeditor" allowOverride="false">
  <system.web>
   <authorization>
    <allow users="bais\canb"/>
    <deny users="*"/>
   </authorization>
  </system.web>
 </location>
```

In db.config the setting *<allowDomainNamesInLoginNames>false</allowDomainNamesInLoginNames>* instructs DesignBais to strip the domain name off the username and send just the login name.

Anonymous Authentication

When using Anonymous Authentication then either Basic or Windows Authentication is also required. DesignBais obtains the *loginPublicUser* from the appropriate entry point within the *db.config* file and reads user record from the DBIUSERS record in the *loginAccount*. The start form for the loginAccount in the Start Account List for this Username will be displayed. It does not have to be the first entry in the list.

This authentication method can be used to display a standard login form for the DesignBais Application. In this setup the *loginUser* may be the only Username that requires a matching Windows username. The standard login form can request the username and password for users defined in the DBIUSERS file and then display the start form for the user.

Refer to *DBALTUSER* for more details on setting up a DesignBais global login form.

In general it is recommend that form based authentication be used for DesignBais.

The login form provides a method for obtaining the user's credentials in order to open their designated start form. This assumes that the login account and the user's start account share the same DBIGLOBAL file. This would normally be the case, but if not then it could have an effect on, for example, whether a user is prompted for Google Two Factor authentication.

Viewing DBIXMLLOG in a terminal session.

DBIXMLLOG entries are of the form:

LIST DBIXMLLOG...

10895*garb*O
10895*garb*I
10886*garb*O
10886*garb*I

Where *garb* represents the weblogon id of the user.

Set up VOC entries to allow you to quickly reset and view the xml log file.

For example set up an entry to clear entries for your weblogon from the DBIXMLLOG file called *CCC*:

```
0001: PQ
0002: HSELECT DBIXMLLOG = "[*your_weblogon*]"
0003: STON
0004: HDELETE DBIXMLLOG
0005: P
0006: HSELECT &COMO& = "[* your_weblogon *]"
0007: STON
0008: HDELETE &COMO&
0009: P
```

Set up an entry to view your DBIXMLLOG entries called say *VVV*:

```
0001: PQ
0002: HSSELECT DBIXMLLOG = "[* your_weblogon *]"
0003: STON
0004: HAE DBIXMLLOG
0005: P
```

When attempting to pin down a problem run the DesignBais application in the browser up to the point just before an error occurs. In your terminal session run *CCC* to clear the xml log.

Then perform the DesignBais function, such as a button click, that causes the error.

In your terminal session run *VVV* to view the xml log entries generated by the button click.

When viewing a long xml string it can be helpful to position focus on the attribute to be viewed. Then use the AE editor features:

CU/^013^010/^253

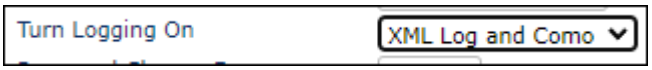
This replaces all carriage return line feed characters with value marks.

Then use the AE Editor EV (edit values) command to view each value as a line within the editor.

Using DBI.RUN.LAST

This program is provided to assist developers to investigate problems encountered in building a DesignBais application.

DBI.RUN.LAST resides in the DBINET file. Source code is provided. The routine is run from a telnet session and will prompt for the input of the user id (WEBLOGON). This is necessary since the DBIXMLLOG file will contain records for all users that have logging set to *XML Log and Como*.



DBI.RUN.LAST requires that the user have logging turned on since the program reads the last generated XML record and passes it to the DesignBais engine BAWEBEXECNET. The output from this is displayed on the telnet screen.

If, for example, the problem is caused by a subroutine not catalogued then running DBI.RUN.LAST will display the error message. Note that your como record may already contain error messages so it is usually best to check your como first.

In order to allow DBI.RUN.LAST to pick up the XML record that contains the error, it is best to run the application up to the point of the error occurring, such as just prior to clicking a button. Then clear all your user records from the DBIXMLLOG file. Then click the button. This will generate the record that DBI.RUN.LAST requires which has a record id of *WEBLOGON*I.*

Refer to the section above Viewing DBIXMLLOG in a terminal session for an explanation of the records in the DBIXMLLOG.

Refreshing the browser

In order to ensure that the most recent version of the DesignBais web and data components are running you must refresh the browser. This clears previous versions from the browser cache.

There are three levels of refresh for the Chrome browser.

The least impactful is simply to open a browser tab and press F5.

A more thorough refresh is obtained by using CTRL-F5.

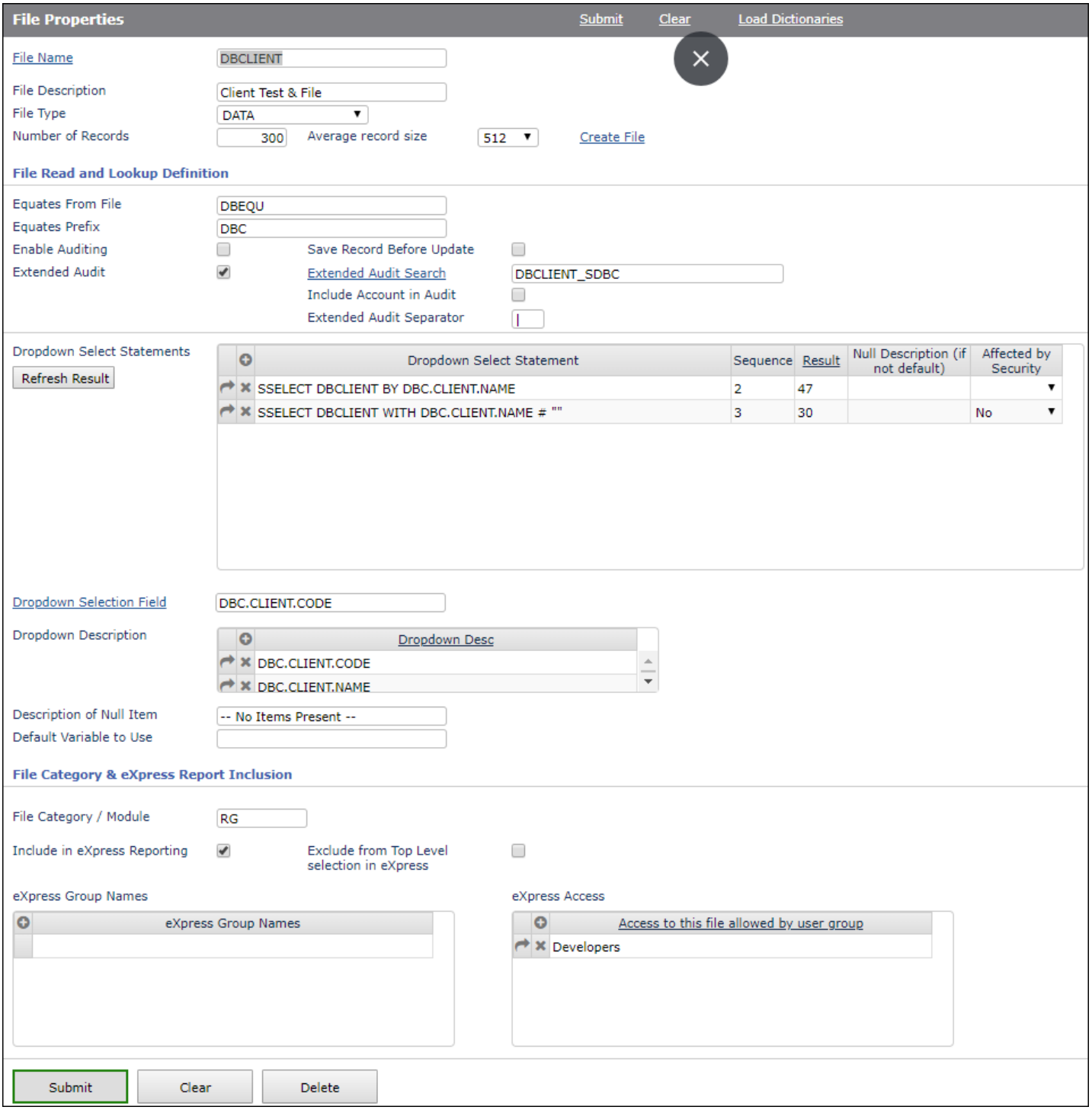
The most impactful refresh is to press F12 and select *Network* from the developer tools menu. Ensure that the *Disable cache* option is ticked, then press CTRL-F5.

Chapter 2 – File Properties

File Properties

Before DesignBais can use a file it is necessary to define it. On the side menu of the DesignBais development environment is the option named **File Properties.**

Records updated in this form update the file **DBIFILES**. The key to this file is the file name.



Prompts

File Name

Enter a name for the file. There can be no underscores or spaces in the filename.

If your application has files with an underscore in the name, it will be necessary to create a File pointer or Q-Pointer to that file with a different name.

Click on the File Name hyperlink to invoke a search for files that are defined to DesignBais.

File Description

Assign a description to the file so that it is easy to identify.

Number of Records

It the file does not exist and you wish to create it, enter the approximate number of records that the file will contain

Average record size

Select an approximate average record size.

Equates From File

Specifies which file to store source-based equates for this file. DesignBais will collate the field properties for each file and create (if required) a source-based Equate definition for the file. An example being:

```
************************************************************************
** Equates for file DBCLIENT
************************************************************************
EQU DBC.CLIENT.CODE TO 0                ;* Client Code
EQU DBC.CLIENT.NAME TO 1                ;* Client Name
EQU DBC.STREETADDRESS TO 2              ;* Street Address
EQU DBC.SUBURB TO 3                     ;* Town/City
EQU DBC.PCODE TO 4                      ;* Zip/Post Code
EQU DBC.STATE TO 5                      ;* State/Province
```

You can see from the above example that each field in the file DBCLIENT has been assigned an attribute reference in the equate definition.

Equates Prefix

Is used to determine a code prefix for each field within the file. This provides for standard naming conventions of fields within a file and also helps to ensure that when including these equate records in your subroutine that you don't get compile errors due to duplicate equate names. This will occur if fields of a file referenced in the subroutine have the same name as fields in another file used in the subroutine. Common examples are files where attribute zero is defined as **ID** and attribute one as **DESC.**

If you assign a prefix to the file then DesignBais would create the equate record as:

EQU *prefix.***ID** TO 0 ; * ID

EQU *prefix.***DESC** TO 1 ; * Desc

Your duplicate definitions are no longer a problem.

Changing the *Equates Prefix* of a file will prompt to regenerate the Program Equates. If *Yes* is selected then the current prefix will be replaced by the new prefix in all field properties.

Enable Auditing

Controls whether modifications made to this file are recorded in the **DBIAUDIT** file. Date, time and user are then recorded for each update to the file.

Save Record Before Update

If auditing is enabled then a copy of the record immediately prior to update will be saved in the **DBIAUDIT** file. Only one version of a record, the most recent, will be saved in the **DBIAUDIT** file.

Extended Audit

This check box turns on extended auditing for the nominated file. The file **DBIAUDIT.EXT** must exist in the account where auditing is being utilized. DesignBais will not create this file automatically. The dictionary level of this file will exist and contains the required DesignBais field definitions.

When extended audit is turned on for a file there will be an item in the DBIAUDIT.EXT with a key of the file being audited. This item holds the attributes that apply to the extended audit function for the file. Another item with a key of DBFILEAUDITCHECK is also created and updated. It holds a multivalued list of file names that have been flagged for extended auditing.

Extended Audit Search

This field is used to define the search process for the nominated file. This search will be used in the audit display form to select records to view audit details.

Include Account in Audit Key

When this check box is checked, the key to the audit record written to DBIAUDIT.EXT will contain the account path. This allows for DBIAUDIT.EXT to be a global file.

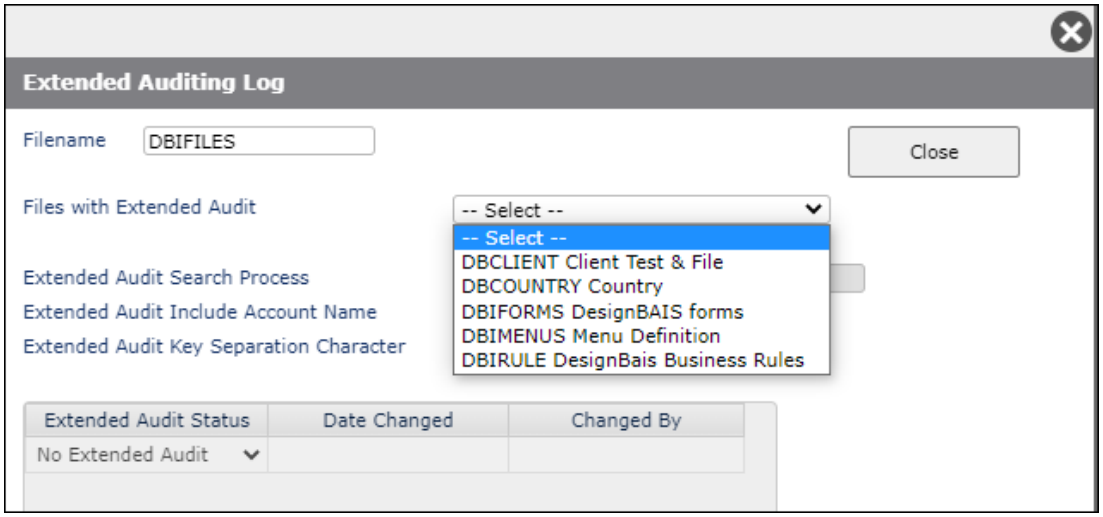
Extended Audit Key Separator

The default separator in the Audit key is a pipe character (|). If the file that is being audited already has pipe characters in the key, it will be necessary to replace the standard separator with another character. Enter the required character in this field. It is important to ensure that the entered character does not appear in any of the keys being written to the audit file.

Extended Audit Status Log

Displays the Extended Audit Status log for the file that is being maintained. This log records when the extended auditing status is changed for a particular file (ie turned on or off). The log records the status (on or off), the date and the user id of the user effecting the change. If the file being maintained in File Properties has never has extended audit set then the log will display a status of *No Extended Audit*.

In the example below the file DBIFILES is shown as never having extended audit. Note that you can select files that have extended auditing status from the dropdown list.



Dropdown Select Statements

Used to provide a standard format for dropdown lists for the file being defined. You can have multiple statements in this field and control which one is used at forms designer stage.

Sequence

Where there are a number of select statements assigned for a file a sequence number can be used to make identification easier. In the Field Properties on a form you can select the FILENAME,SEQUENCE combination for a dropdown list.

Result

This will populate with the number of items that are selected by the corresponding Dropdown Select Statement. It can be refreshed by highlighting the required statement row and clicking the Result search label.

Null Description if not default

Enter the description for the null item if this sentence does not use the default Null Description field.

Affected by Security

This flag determines is the list created is affected by Entity Security Settings. Typically file-based lists are not governed by any security. In some instances you may wish to change the contents of the file-based list to show only those items that the user has access to. Please see the chapter on **Entity Based Security** later in this document.

Dropdown Selection Field

Used to determine which field on the file being defined is used to provide the key for dropdown lists. When the end user selects a description in a dropdown list, the code associated with the description will be the field define here.

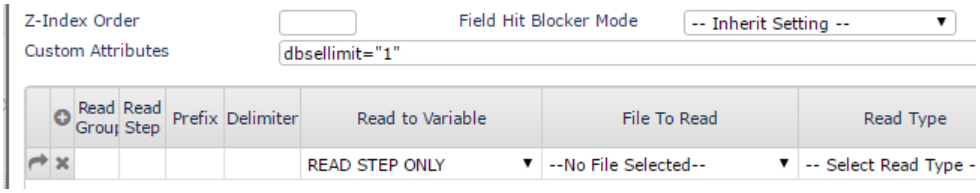
Dropdown Description

Used to define the description fields that are to make up the details within a dropdown list. This helps to provide meaningful descriptions to codes in a field prompt. You can use more than one field to make up a dropdown list description. If more than one field is used, the resulting description with have the fields concatenated with a space in between.

Note that you can use the Custom Attribute dbsellimit="1" (refer to Input Field Properties in Forms Designer) to limit the width of the field that the browser renders. The width will be derived from the Display Width Pixels of Fields to Display, or the Select Field Width of the Entry Supplied grid in a Selection Process, or the field length as defined in the Field Properties.

The width calculation adds all field lengths and multiplies by 7 where 7 is an approximation for converting average font character widths to pixels. So for 2 fields with length of 20 the calculation is:

(20 + 20) * 7 = 280 pixels.



Description of Null Item

Used to define the description for an unselected dropdown list. The DesignBais user interface is event driven. If you wish to trigger a server event off a dropdown list, then you need to have a null item description. If the first value in the list is the item that the user wishes to be selected and you do not have a null item description then the triggered event will not fire because there would be no change.

Default Variable to Use

Allows for the assignment of one of the one hundred standard DesignBais variables to store records read from this file. The choices are:

DBRECORD
DBOTHER.RECORD(1) to DBOTHER.RECORD(99)

The value in this field will then be used whenever you add a field from the defined file to the Forms Designer

File Category / Module

This will identify the file as belonging to a module and provides the developer with assistance to select files belonging to a module.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Include in eXpress Reporting

This field determines whether the file being defined is available to the eXpress reporting interface. If you clear this flag you will be prompted to also clear the field property flag *Include this Field in eXpress Reporting* for all field properties for this file.

Exclude from Top Level selection in eXpress

If this check box is checked, the filename will be excluded from the top-level reporting menu in DesignBais eXpress. The file will still be available as a sub-table selection if provided by another file.

eXpress Group Names

Fields that are presented in the eXpress report writer can be grouped so that it is easier for the end-user to find fields that contain related data.

The Group Name define a name that groups the fields available within the eXpress report writer for a file.

This is positional, so it is recommended that you do not delete, or inserts rows in this table.

eXpress Access

You may nominate the User Groups that can access a file/table in the eXpress Report Writer. If there are no groups defined, then all users will have access to the file within the eXpress report writer.

Buttons

Create File

Is used to create the file. This does not update the file properties record. You must also press the submit button to define the file to DesignBais.

Submit

Updates the file properties record. This makes the file available to DesignBais.

Clear

Clears the form without updating the file properties.

Delete

Delete the file definition record from the **DBIFILES** file. The file can no longer be accessed by DesignBais.

Accessing DesignBais Files from within DesignBais Applications

Developers may want to access DesignBais files within their applications.

This can be done by creating a file pointer, or Q pointer, with a name that does not begin with “DBI”, that points to the DesignBais file.

Why are Q pointers required

It is necessary to create Q pointers in order to access DesignBais files from within applications built using DesignBais. DesignBais uses four DBISYS… files to hold the forms, field properties, selects, reports, menus and other required items relating to Designbais tools.

These files are:

DBISYSFILES contains all DesignBais File Properties
DBISYSFORMS contains all DesignBais Forms, Reports, Menus, Style Groups, Styles and some other parameters
DBISYSPROP contains all DesignBais Field Properties
DBISYSSELECT contains all DesignBais Selection Processes

DBIFILES and DBISYSFILES contain details about files defined in DesignBais. They also hold multivalued lists of all field property names defined for each file. When a form is rendered by the browser DesignBais uses the list of field property names on DBIFILES or DBISYSFILES rather than accessing each field name by a read of the DBIPROP or DBISYSPROP file. This reduces the time taken to render and process DesignBais forms.

DesignBais attempts to read the form file properties record for a form, first from DBIFILES, and if not found, then from DBISYSFILES. DesignBais then utilizes the list of field properties that it finds on this file properties record.

In the case of DesignBais tools form, if a record is found on DBIFILES for a form such as, for example, DBIFILES*D10, then DesignBais will attempt to access the fields listed on this record, rather than using the correct DBIFILES File Properties record held on DBISYSFILES.

This leads to a number of problems such as missing fields or incorrect field properties. If the record on your local DBIFILES file is null (empty) then DesignBais will not find any field properties which can lead to the browser rendering a blank page, and the developer wondering what has happened!

Example showing the steps required to set up a pointer to DBIUSERS

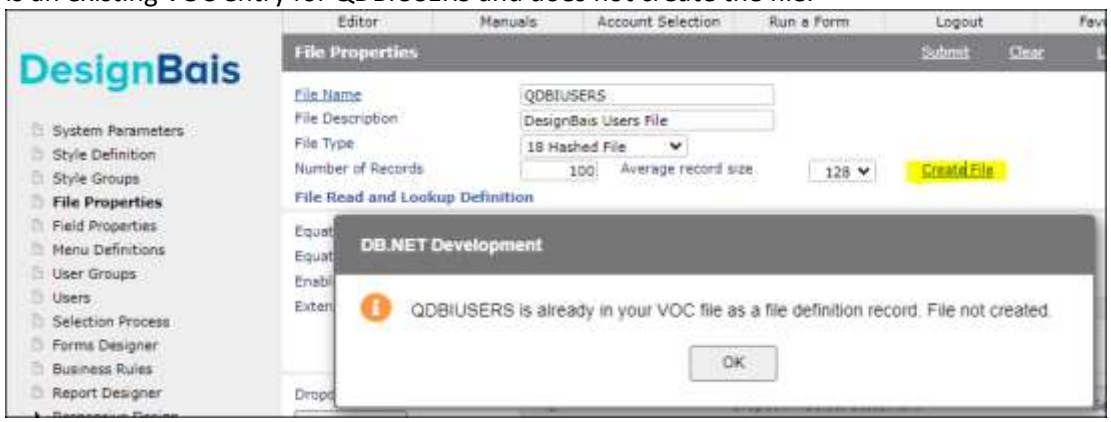
For example a developer can set up access to the DesignBais Users file DBIUSERS using the following steps.

In the account where your application runs create a Q pointer called, for example, QDBIUSERS.

On UniVerse the pointer to QDBIUSERS can will take the form shown to the right.
Essentially you can copy the DBIUSERS VOC entry and change attribute 1 from “F” to “Q”.
The data level points to the DesignBais users file DBIUSERS.
The dictionary level points to the dictionaries which, except for D3, reside in the DBINET account.

```
>ED VOC QDBIUSERS
3 lines long.

----: P
0001: F
0002: DBIUSERS
0003: E:\HOME\DESIGNBAIS\DBINET\D_DBIUSERS
```

After creating the QDBIUSERS VOC entry you can then set up the File Properties record in DBIFILES.
This is shown in the snip below which demonstrates that if you attempt to *Create File*, DesignBais recognizes that there is an existing VOC entry for QDBIUSERS and does not create the file.



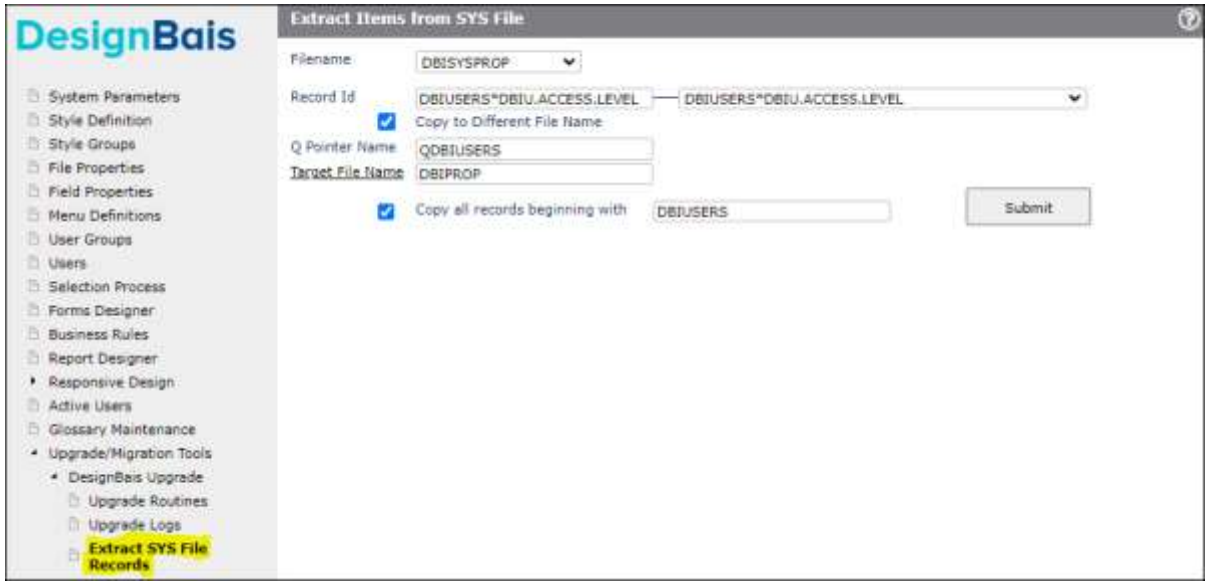
In the File Properties form (DBIFILES_D10), shown above, input the name “DBINET” in the *Equates From File* field. Input the DBIUSERS equates prefix “DBIU” in the *Equates Prefix* field. Then submit the record. This will create a record on DBIFILES with a key of QDBIUSERS.

You then need to copy all DBIUSERS field properties from DBISYSPROP into DBIPROP. There is a tool to do this which is accessed from the DesignBais Upgrade menu, called *Extract SYS File Records*, and is shown below.

Field properties are held in DBISYSPROP so select this file name from the dropdown in the source filename field.

Enter any existing DBIUSERS field property name in the *Record Id* field. In the example below we have entered “DBIUSERS*DBIU.ACCESS.LEVEL”.

Alternatively you can select a field property from the dropdown selection list.



You must check the box *Copy to Different File Name*. This opens up the *Q Pointer Name* field. Enter the name of your local file that points to DBIUSERS which is, in our example, QDBIUSERS.

The *Target File Name* is DBIPROP. The field properties for your local DBIFILES record called QDBIUSERS must exist in the local DBIPROP file.

Finally check the box *Copy all records beginning with*. This instructs the tool to select all DBISYSPROP records with names that commence with “DBIUSERS*” and copy them to DBIPROP, changing the name from DBIUSERS*fieldname to QDBIUSERS*fieldname.

It is then imperative that you enter your file name QDBIUSERS in the File Properties form DBIFILES_D10 and click the Submit button. This dummy amend procedure will update the DBIFILES record with the list of all field properties for the file.

You are now ready to use your local DBIFILES record called QDBIUSERS within your DesignBais application.

Upgrading to a new Designbais Release

When you upgrade to a new version of DesignBais you will have to repeat the *Extract SYS File Records* option in order to incorporate new and changed field property records into your local DBIPROP file. You will also need to dummy amend the file QDBIFILES in your File Properties form.c

Chapter 3 – Field Properties

Field Properties

Before you can use a field in DesignBais it must be defined in field properties. The field properties form provides an extended dictionary for the fields within a file.

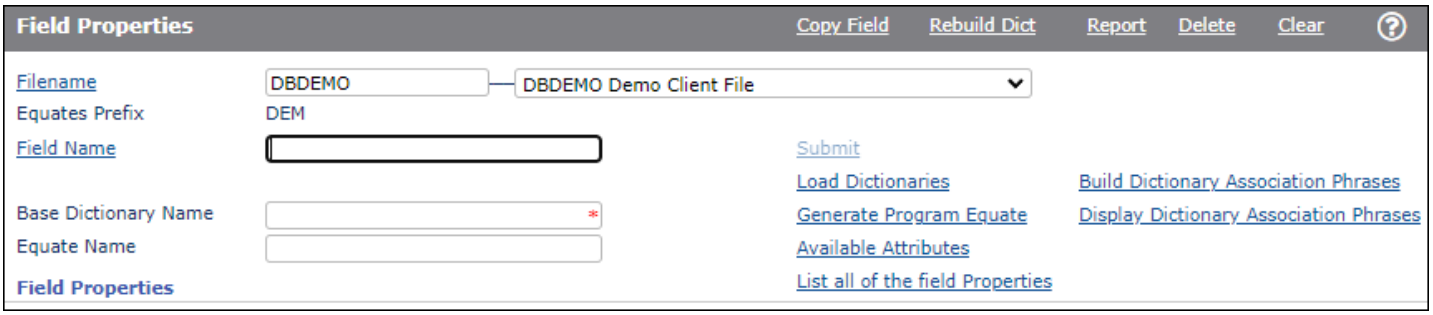
Records updated in this form update the file **DBIPROP**. The key to this file is:

Filename * Field Name. Eg. DBDEMO*DEM.COUNTRY

DesignBais maintains an internal index of all fields that belong to file. This is done to ensure that performance is always optimal. It also allows for the extended use of glossaries for field names.

There are a number of sections to the field properties form. In this document they have been separated for simplicity.

Filename and Field Name



Prompts

Filename	Enter a name for a file that is known to DesignBais. You may also use the dropdown list to select a filename. Click on the Filename hyperlink to invoke a search for files that are defined to DesignBais.
Field Name	This field represents the name of the field within DesignBais. This does not necessarily represent the name of the dictionary definition. You may decide to name your field in DesignBais as DEM.COUNTRY but have the dictionary to the field named COUNTRY. Click on the Field Name hyperlink to invoke a search of field for the selected file.
Base Dictionary Name	Allows you to create a dictionary definition with a different name than the field property name. This will allow you to keep your dictionary names simple, whilst maintaining a prefix in the field property to avoid duplicate names if you are using program equates. This field is not active in version v4.1.4 and prior.
Equate Name	Used to define the name for the field that will be used in program equate records. This will help remove the issue of duplicate equate names in your source programs.

Buttons

Submit	Will update the field property record on the DBIPROP file. See the descriptions for the System Parameters fields “Routine to Call to Verify Dictionary” and “Always Update Base Dictionary”.
Load Dictionaries	Will load the existing dictionaries from the file. This will significantly speed up any conversion process. If SB+ extended dictionaries are present then these will be given preference over the normal database dictionary records.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

You may press this button at any time. DesignBais dictionaries will not be overwritten. This helps to keep DesignBais field properties in sync if you are still performing development using another tool.

Before the file can be used it is necessary to view one of the fields loaded and Submit it. This will re-generate DesignBais indexes for the file.

See the description for the System Parameters field “Routine to Call to Verify Dictionary”.

Generate Program Equate

Is used to create program equate definitions for the current file. These equates will be written to the file described in the File Properties record for the file.

DesignBais will create an Equate item named **E.***filename.*

These items can be included in your source using the **INCLUDE** statement.

Available Attributes — Use this option to check which attributes of a file are defined as a field property and/or populated with data.

List all of the field Properties

Can be used to generate a listing of all of the Field Properties for the current file. The report produced will be displayed at the bottom of the Field Properties form.

You may select and load any field in the list by clicking on the field name in the first column. If the number of fields is greater than the form length a scroll-bar will appear on the right of the form.

Field Name	Screen Label	Attrib	Type	eXpress Reporting	Length	Dec	Just	Conv	Mult	Work	Group
DBIGO.GLOBAL.TO.EMAIL	Email To Address	1	A	No	40		L		N	N	
DBIGO.GH.DEFAULT.FORM	Default Header Form	1	A	No	30		L		N	N	
DBIGO.GF.DEFAULT.FORM	Default Footer Form	1	A	No	30		L		N	N	
DBIGO.DBSTORE.LMF.ATTR	Last Main Form	1	A	No	30		L		N	N	
DBIGO.DATE.FORMAT	Date Format	1	A	No	20		L		N	N	
DBIGO.CF.FONT	Font Family	1	A	No	20		L		Y	N	FONTFACE
DBIGO.CENTER.ALL.FORMS	Center All Forms	1	A	No	3		L		N	N	
DBIGO.GEN.KEY.HICHART.WK	HICHART	1.1	A	N/A	20		L		N	Y	
DBIGO.GEN.KEY.LOGSIZE.WK	LOGSIZE	1.2	A	N/A	20		L		N	Y	
DBIGO.GEN.KEY.DATE.WK	DATE	1.3	A	N/A	20		L		N	Y	
TEST.FIELD	Test	1.3.4	A	No	20		L		N	N	
DBIGO.GEN.KEY.SCRIPTS.WK	SCRIPTS	1.4	A	N/A	20		L		N	Y	
DBIGO.GEN.KEY.META.WK	META	1.5	A	N/A	20		L		N	Y	
DBIGO.GEN.KEY.LINKS.WK	LINKS	1.6	A	N/A	20		L		N	Y	
DBIGO.GEN.KEY.GLEM.WK	Global Email	1.7	A	N/A	20		L		N	Y	
DBIGO.GEN.KEY.LOGOUTURL.WK	Logout URL	1.8	A	N/A	20		L		N	Y	
DBIGO.GEN.KEY.READWRITE.WK	READWRITE	1.10	A	N/A	20		L		N	Y	
DBIGO.GEN.KEY.GLOSSARY.WK	GLOSSARY	1.11	A	N/A	20		L		N	Y	
DBIGO.GEN.KEY.WEBSERVICE.WK	WEBSERVICE	1.12	A	N/A	20		L		N	Y	
DBIGO.PASSWORD.WK	Password	2	A	N/A	20		L		N	Y	
DBIGO.SNS.CONFIGNAME	Config Parameter	2	A	No	20		L		N	N	
DBIGO.RESIZE.MAXIMUM	Maximum Width	2	N	No	5		R		N	N	
DBIGO.PRINTOPTIONS	Disabled Print Options	2	A	No	30		L		Y	N	

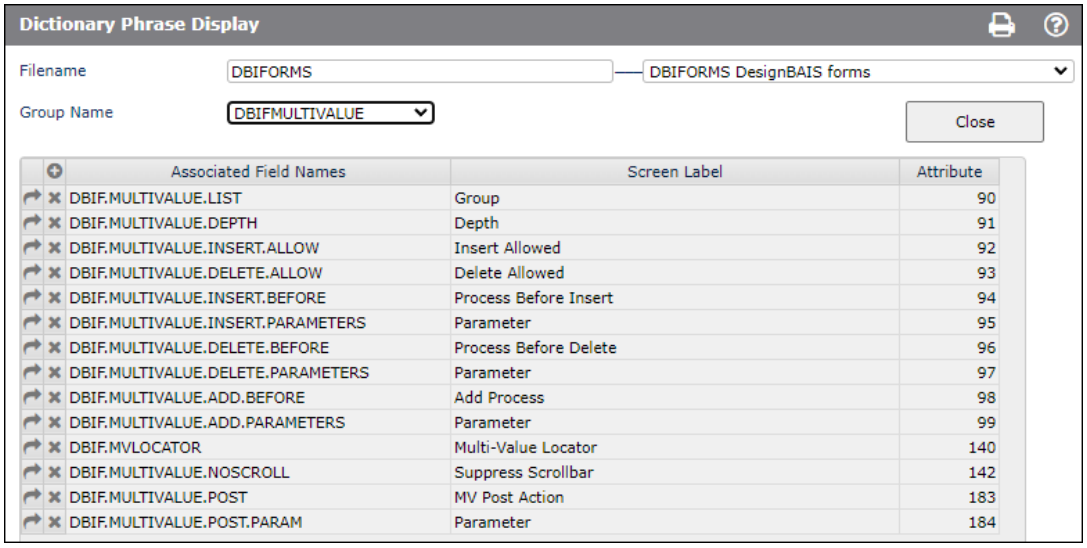
The *Attrib* column displays the attribute position and, if present, the multivalue and subvalue positions separated by a period (“.”).

Build Dictionary Association Phrases

Builds *PH-type* dictionary items for each set of associated multivalue fields in the file. These phrase items have *PH* in attribute 1 and a space-separated list of fields that share the same Form Group name. The id of the phrase item is the Form Group name.

Display Dictionary Association Phrases

Displays the list of dictionary names contained in the phrase item.

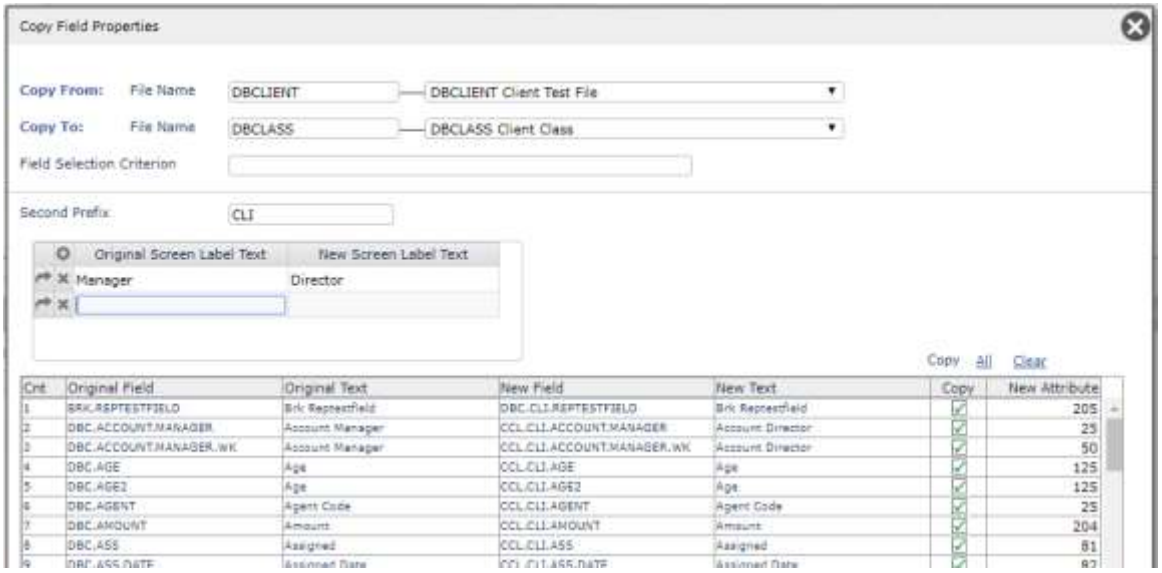


Associated Field Names	Screen Label	Attribute
DBIF.MULTIVALUE.LIST	Group	90
DBIF.MULTIVALUE.DEPTH	Depth	91
DBIF.MULTIVALUE.INSERT.ALLOW	Insert Allowed	92
DBIF.MULTIVALUE.DELETE.ALLOW	Delete Allowed	93
DBIF.MULTIVALUE.INSERT.BEFORE	Process Before Insert	94
DBIF.MULTIVALUE.INSERT.PARAMETERS	Parameter	95
DBIF.MULTIVALUE.DELETE.BEFORE	Process Before Delete	96
DBIF.MULTIVALUE.DELETE.PARAMETERS	Parameter	97
DBIF.MULTIVALUE.ADD.BEFORE	Add Process	98
DBIF.MULTIVALUE.ADD.PARAMETERS	Parameter	99
DBIF.MVLOCATOR	Multi-Value Locator	140
DBIF.MULTIVALUE.NOSCROLL	Suppress Scrollbar	142
DBIF.MULTIVALUE.POST	MV Post Action	183
DBIF.MULTIVALUE.POST.PARAM	Parameter	184

Header Bar Buttons

Copy Field — Provides the ability to copy the currently displayed field to another name.

The Advanced Copy button provides a more comprehensive copy function allowing selected fields on one file to be copied to another file, with the file *Equates Prefix* for all copied fields changed to that of the target file. In addition a second prefix can be added in order to create a group of fields on the target file. The attribute on the target file can be amended. In the example below fields from DBCLIENT are being copied to another file DBCLASS. The equates prefix is changed from DBC to CCL and a second prefix of CLI is being inserted into all field property names in order to group them. If Screen Label text contains the string *Manager* then it is replaced with *Director*. Fields can be deselected so that they are not part of the copy.



Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Rebuild Dictionary — Will re-build the contents of the dictionary from the items contained in the field properties file DBIPROP. This feature will make updating releases of your software much easier.
See the description for the System Parameters field “Routine to Call to Verify Dictionary”.

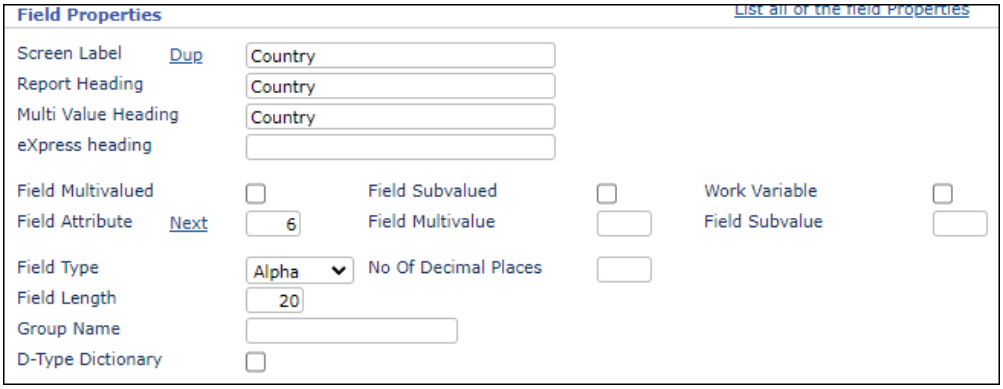
Report — Will produce a report of all Field Properties for the selected file.

Field Name	Screen Label	Attr	MV No	SV No	Type	Len	Dec	Just	Out Conv	MV	Wk	Select Conv	Group Extract	Group	Lookup File
DBIGO.CENTER.ALL.FORMS	Center All Forms	1			A	3		L		N	N				
DBIGO.CF.FONT	Font Family	1			A	20		L		Y	N			FONTFACE	
DBIGO.DATE.FORMAT	Date Format	1			A	20		L		N	N				
DBIGO.DBSTORE.LMF.ATTR	Last Main Form	1			A	30		L		N	N				
DBIGO.GF.DEFAULT.FORM	Default Footer Form	1			A	30		L		N	N				
DBIGO.GH.DEFAULT.FORM	Default Header Form	1			A	30		L		N	N				
DBIGO.GLOBAL.TO.EMAIL	Email To Address	1			A	40		L		N	N				
DBIGO.HCHART.THEME	Highchart Theme	1			A	25		L		N	N				
DBIGO.LINKS	Custom CSS Links	1			A	10		L		Y	N				
DBIGO.LOGIN.IMAGE	Login Image	1			A	20		L		N	N				
DBIGO.LOGOUTURL	Logout URL	1			A	60		L		Y	N				
DBIGO.LOGSIZE	Log File Size	1			N	10		L		N	N				
DBIGO.META	Meta Details	1			A	10		L		Y	N				
DBIGO.READWRITE	READWRITE Subroutine	1			A	60		L		N	N				
DBIGO.RESIZE.PROGRAM	Resize Subroutine	1			A	25		L		N	N				
DBIGO.SCRIPTS	Script References	1			A	10		L		Y	N				
DBIGO.SNS.URL	SNS Web Service URL	1			A	50		L		N	N				
DBIGO.TRACK.GLOSSARY	Track Glossary Usage	1			A	10		L		N	N				
DBIGO.WEBSERVICE	Web Service Subroutine	1			A	10		L		N	N				
TEST.FIELD	Test	1	3	4	A	20		L		N	N				
DBIGO.CF.SOURCE	Source	2			A	20		L		Y	N			FONTFACE	
DBIGO.GF.ACCOUNTS	Accounts	2			A	30		L		Y	N				
DBIGO.GH.ACCOUNTS	Accounts	2			A	30		L		Y	N				
DBIGO.GLOBAL.FROM.EMAIL	Email From Address	2			A	40		L		N	N				
DBIGO.PASSWORD.MINIMUM	Minimum Password Length	2			N	2		L		N	N				
DBIGO.PRINTOPTIONS	Disabled Print Options	2			A	30		L		Y	N				
DBIGO.RESIZE.MAXIMUM	Maximum Width	2			N	5		R		N	N				
DBIGO.SNS.CONFIGNAME	Config Parameter	2			A	20		L		N	N				
DBIGO.ENCODE.HTML	Encode HTML	3			A	3		L		N	N				
DBIGO.GF.ACCOUNT.DEFAULTS	Account Default Footer	3			A	30		L		Y	N				
DBIGO.GH.ACCOUNT.DEFAULTS	Account Default Header	3			A	30		L		Y	N				
DBIGO.PASSWORD.MCASE	Mixed Case Mandatory	3			A	1		L		N	N				
DBIGO.RESIZE.MINIMUM	Minimum Width	3			N	5		R		N	N				

Delete — Delete the currently displayed field property record.

Clear — Clear the form display.

Field Properties



Prompts

Screen Label	The default text to be used for the field when added to a DesignBais form. DesignBais parses the Field Name to arrive at a default Screen Label. This can be amended or overtyped. DesignBais sets a default Field Type, Field Length, and Output Justify based on the name of the field. Common delimiters used in this process are '._$][-!|' Thus field names which include the string delimiter:'DATE' or delimiter:'TIME' (where delimiter is one of the common delimiters listed here) will set the type to 'Date' or 'Time' respectively. The string 'AMT' or 'AMOUNT' causes a 'Numeric' default type and sets the Output Justify to 'Right'. Use the "Dup" button to duplicate the Screen Label in the Report and Multi Value Heading fields.
Report Heading	The default text to be used for the field when added to a DesignBais report
Multi Value Heading	The default text to be used for the heading in a multivalue grid. DesignBais will automatically wrap multivalue headings. This allows you to provide meaningful descriptions for multivalue fields.
eXpress Heading	By default eXpress uses the Report Heading on reports. This field is used to provide an alternative heading for eXpress reports.
Field Multivalued	This check box is used to define the field as multivalued. This does not refer to a field that resides in a multivalue position, Eg. Attr<6,5>, but a field that occupies the entire attribute. If you want to define a multivalue position leave this check box un-checked. If this check box is ticked (ie the field is flagged as multivalued) or the field has Group Name, then DesignBais will perform mutlivalue type reads on this field. This means that a read will be performed on each row of the multivalue grid in which the field occurs. If the field is placed on a form with type other than Multivalue Input or Output then the form may perform other than as expected.
Field Subvalued	DesignBais supports sub-value text fields within a multivalue. These are very useful to add text descriptions associated with other multivalue fields. This does not create a sub-valued association. A field defined as a sub-value will be displayed within a text area field in the multivalue grid. This will allow the user to type free text against the multivalue. When the user presses a [Return] a sub-value will be added to the record.
Work Variable	This check box indicates that the field is to be stored (by default) in the DesignBais common variable named **DBWORK**. The field will not be included in the main record read. Work variables can be used to add fields to a form or a report that do not affect a record read from a file. These fields are generally used to display the results of calculations. Fields assigned a DBWORK variable should be used for the key elements in multi-part keys.
Field Attribute	Refers to the attribute position of the field within the record. Use the "*Next*" button to use the next available attribute position for the file. Note that gaps in the range of defined attributes are ignored. The value returned will be the greatest defined value + 1. This button discriminates between Work fields and actual fields and only returns a value if the Field Attribute is null.
Field Multivalue	Defines the multivalue position of the field within the attribute. You should not have a value in this field if the **Field Multivalued** is checked
Field Subvalue	Defines the sub-value position within the multi-value. This field should be blank if the **Field Subvalued** field is checked. This allows you to create an Attribute, Multivalue and Subvalue reference for a field.
Field Type	Defines the standard conversion and validation type to be used for the field. DesignBais performs simple validation for field type on the client-side, reducing the requirement for server validation processes.

Date and time formats are controlled by the client computers' Regional and Language settings. The standard formatting of numeric fields are controlled by the value in the No of Decimal Places field.

Number of Decimal Places Defines how numeric fields are formatted as standard.

If this field is left blank numeric fields are not formatted, but they are validated. When blank the numeric field cannot contain negative numbers.

If a number (including zero) is entered, the field is formatted with the numeric format character. If the number of decimal places is greater than zero then decimal places are added.

Eg.	123456	Negative number -123456
Decimal Places = blank	Result 123456	Not valid
Decimal Places = 0	Result 123,456	Result -123,456
Decimal Places = 2	Result 123,456.00	Result -123456.00

Number of Decimal Places must have a value if the numeric field is to allow for negative numbers.

Field Length

Defines the default length of the field. This controls the length of the input or output field when placed on a form or report. This does not indicate the maximum number of characters that can be entered within a field. The DesignBais forms designer has a check box assigned to this task.

Group Name

Multivalue fields are often collected into associated groups. An example might be contacts associated with a client. The contact multivalues may have fields that include contact name, contact email address, contact phone number, contact fax and contact notes.

The **Group Name** field allows the designer to group these fields by a common name. For contacts for example, the group name may be CONTACT. The advantage of applying a Group Name to a set of associated fields is that DesignBais will manage insertion and deletion of the all associated fields whether or not all the associated fields appear on a form.

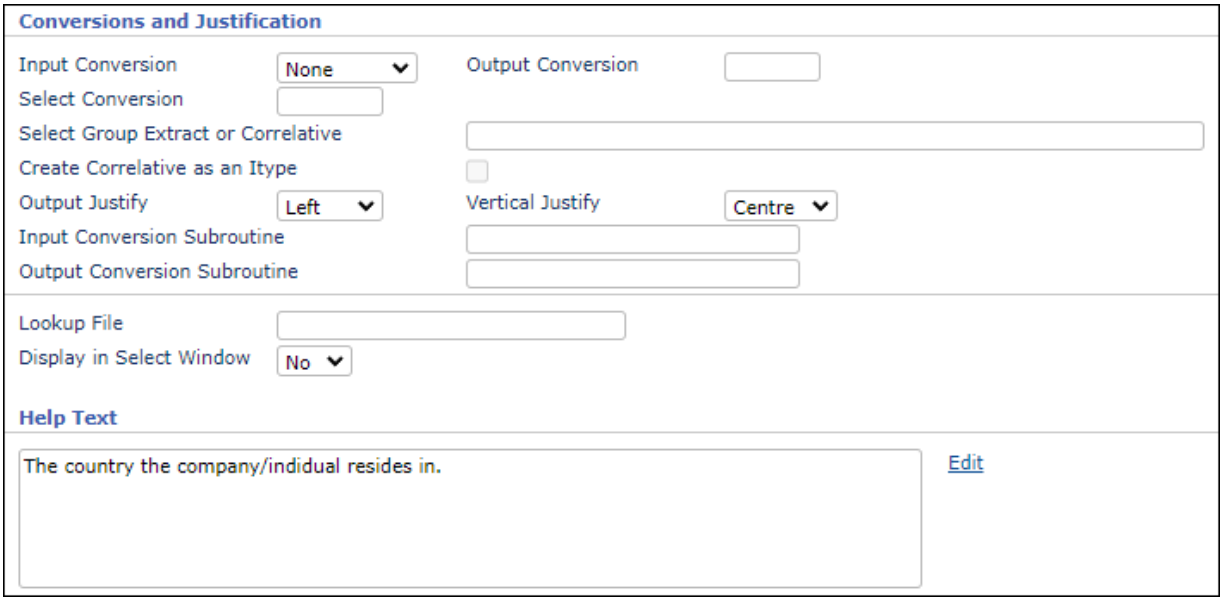
Fields in a multivalue associated grid on a form, and all associated fields not on the form, must all have the same Group Name if the values in these fields are to remain associated.

- A Group Name must only contain alpha and numeric characters
- Group Names must not contain a dot (.)
- If a Group Name includes the string “ASSOC” then DesignBais will NOT maintain the association of multivalues in fields that are not on the form.

D-type Dictionary

Check this field to create the field dictionary item as a D-Type rather than an A-Type. On a Universe database D-Type dictionaries are required for Universe EDA applications. D-Types are forced for UniData. A-Types are created for D3.

Conversions and Justification



Prompts

Input Conversion

Standard **Input Conversion** to convert alpha text fields to upper and lowercase. This conversion is performed on the client to avoid a server call.

Output Conversion

Is used for reporting purposes only. The output conversion uses any conversion that will work within the context of the **OCONV()** function. Examples MD2, MR22, MTS, MCU.

Note that this output conversion is ignored on “Date” type fields. Output conversion of dates is determined by the date format set in System Parameters or in Global Parameters.

Select Conversion

Is used to provide conversions of a field within select statements. This allows you to ensure that fields stored in multi-case formats can be found easily within selections. By default search text will be case sensitive. The select conversion mask entered here will override this default behaviour.

Eg. Client Name may be stored as entered [Joe Bloggs Inc]

The end user when searching for "Joe Bloggs Inc" would need to enter "Bloggs" to locate the record.

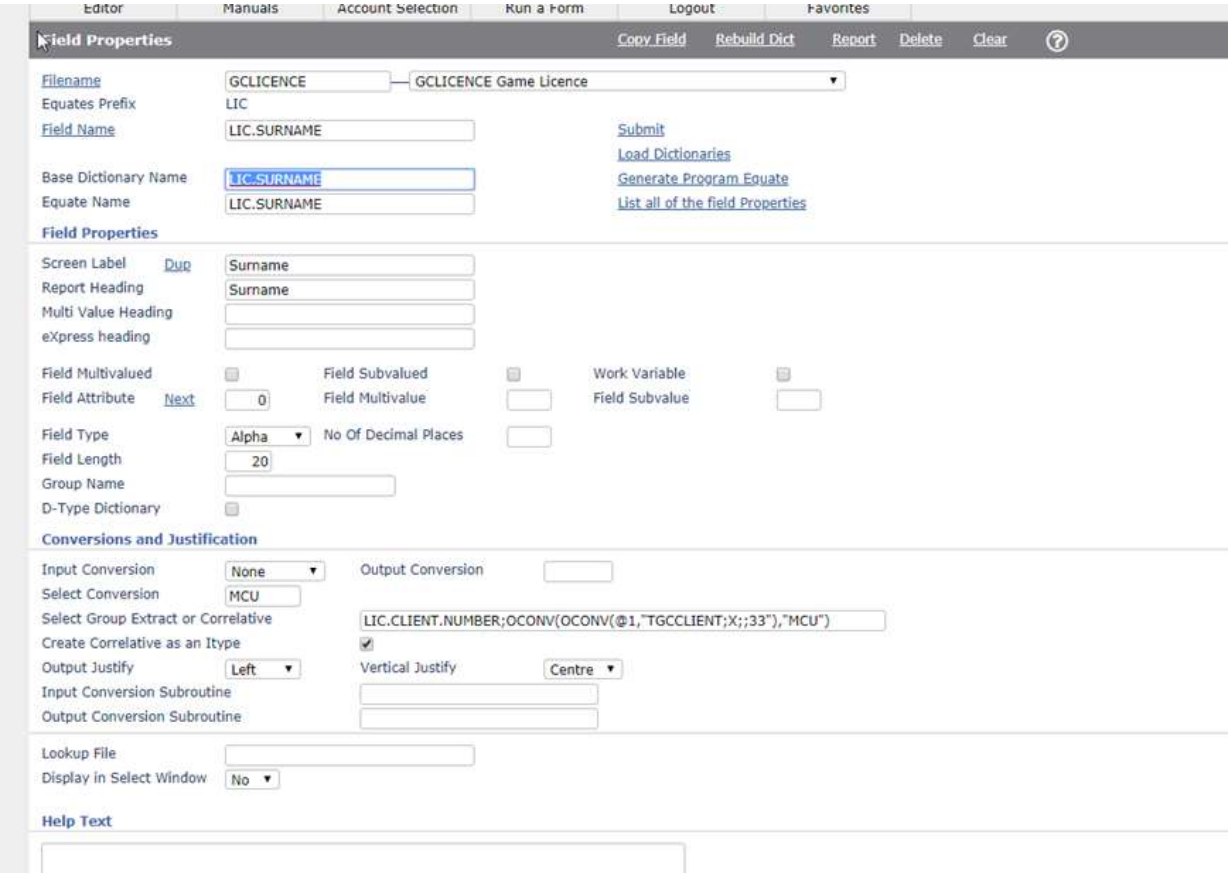
Assigning a **Select Conversion** of MCU will create another dictionary definition named *FieldName*_s. This is only created, however, if the Routine to Call to Verify Dictionary allows the dictionary to be updated. The '_s' indicates that the dictionary definition is only used for selection/searching purposes. This allows the user to enter "bloggs" and still find the record. The resultant selection will match the case of the user entry with the case assigned in the _s definition.

The resultant selection would be: SSELECT DBCLIENT WITH DBC.CLIENT.NAME_s = "[BLOGGS]", though the user entered "bloggs"

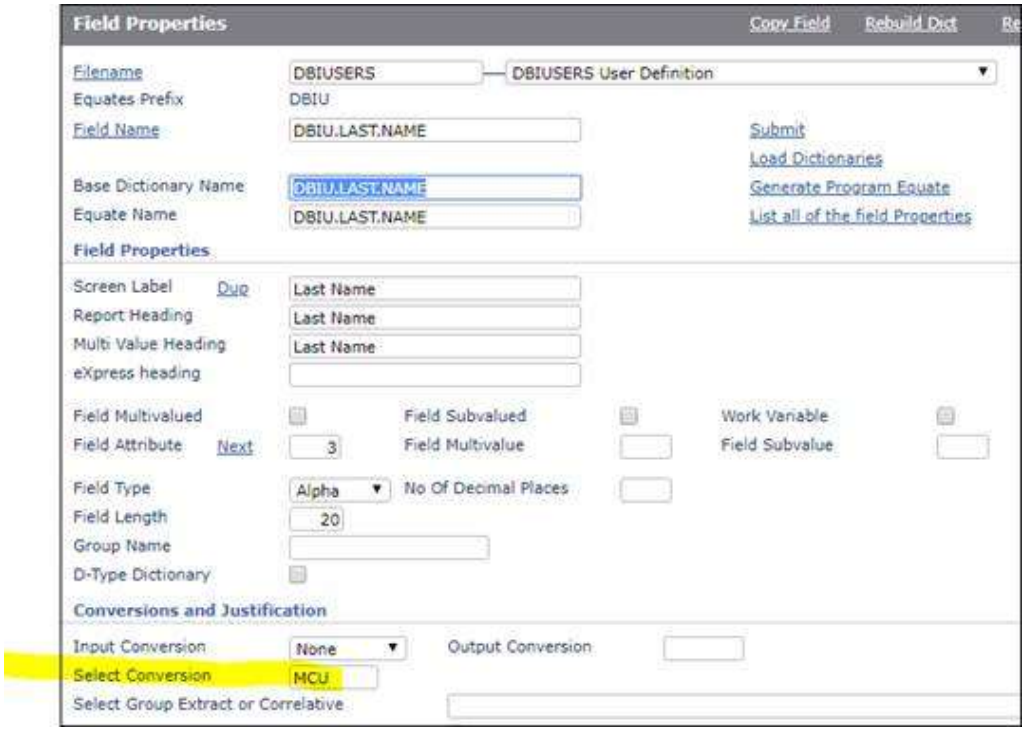
The '_s' dictionary definition contains the MCU conversion.

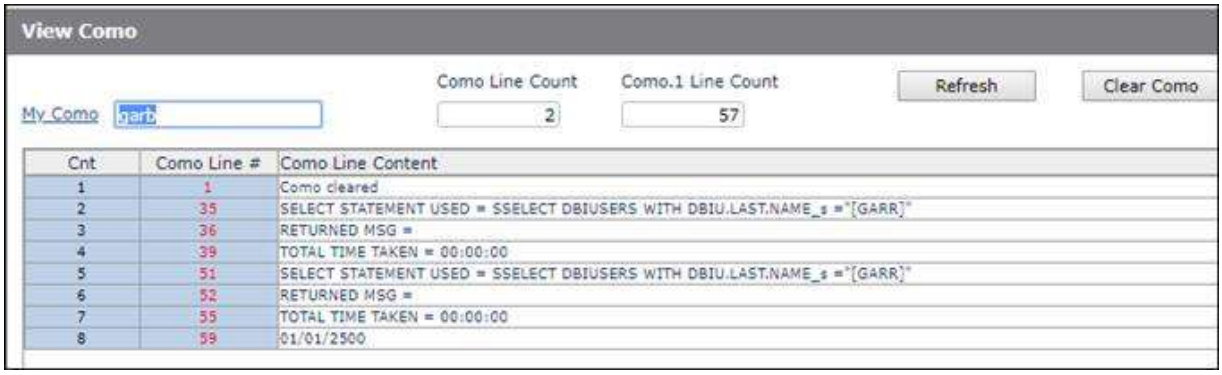
The designer of the selection process does not need to assign the '_s' definition in the selection process. If one exists DesignBais will include it automatically.

The following example shows the syntax, on Universe, for setting up a selection field LIC.SURNAME that points to an attribute on another file. In this case the Client Name being used for the case-insensitive selection resides in attribute 33 of the GCCLIENT file, and the selection process is defined on the GCLICENCE file.

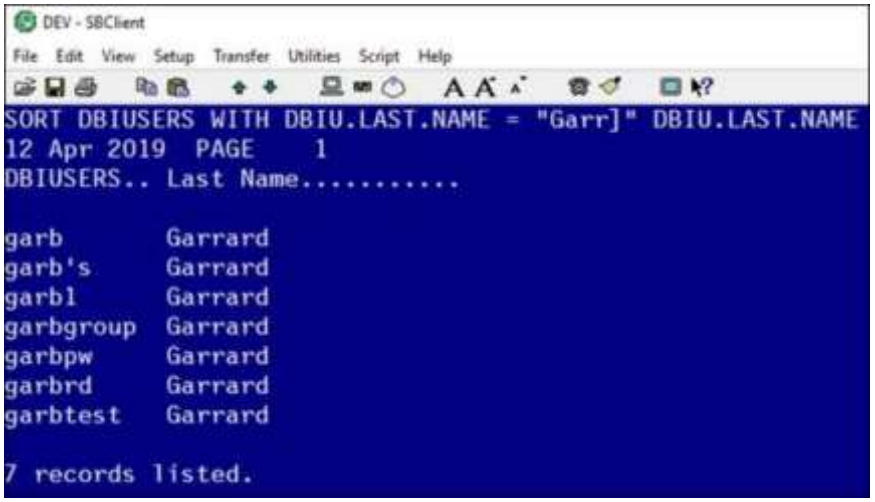


Here is an example of the use of a field property *Select Conversion* of *MCU* for the DBIU.LAST.NAME field on DBIUSERS file. Entering the *MCU* select conversion triggers the creation by DesignBais of the *DBIU.LAST.NAME_s* dictionary item. This is the field that is used in the selection statement as shown below.



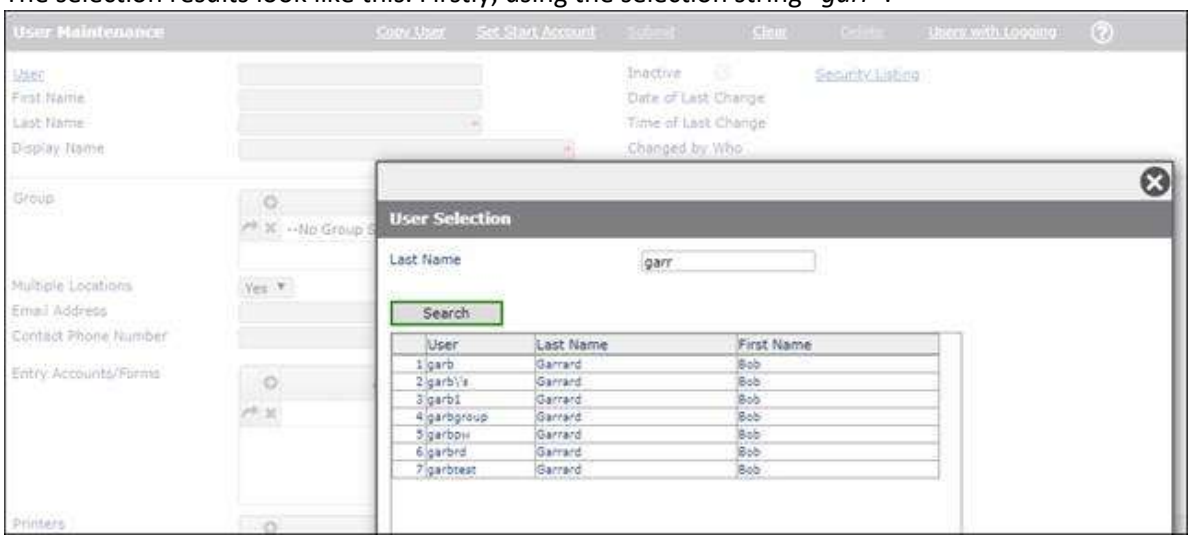


The data set looks like this:

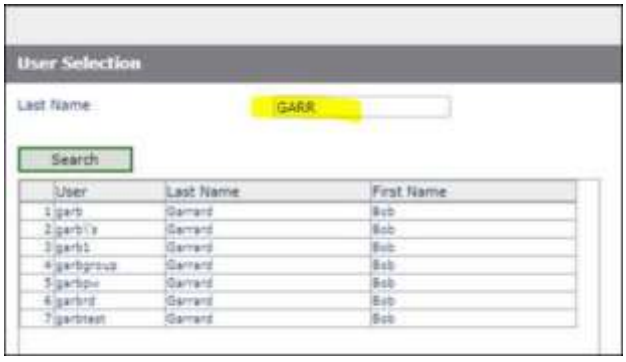


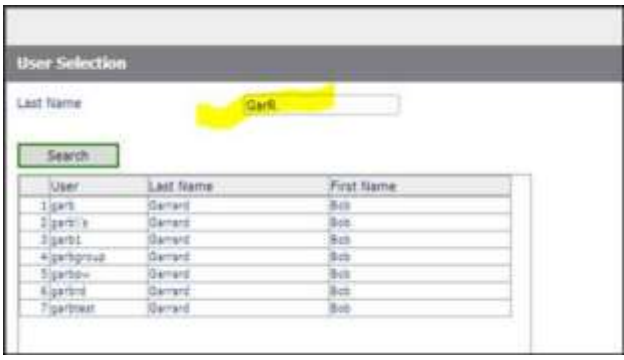
Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

The selection results look like this. Firstly, using the selection string "*garr*":



Using selection strings "*GARR*" or "*GarR*" yield the same results:





Select Group Extract or Correlative

In some cases it may be necessary to create a **Group Extract or Correlative** for the purpose of selections. You can enter the group extract or correlative at this point.

If the field contains a multi-segmented element, delimited by a character, enter the group extraction in the normal form of G0*1.

Eg.

The file ORDERS.DETAIL has a multi-part key of ORDER.NUMER * LINE.NUMBER. To build the extract for LINE.NUMBER you would enter G1*1. To derive ORDER.NUMBER the value would be G0*1.
If the starting character within this field is not a 'G' (group extract) then it is assumed that the expression is in a correlative format.

Eg. Tmyfile;X1;;2 - F;16;(Tmyfile;X;;1).

Correlative functionality may vary depending on the database platform.

Create Correlative as an IType

For DBMS platforms that support I-Descriptors (I-Types), this check box allows dictionary definitions to be written as an IType rather than as a "Pick Style" A or S type. This check box is significant even if the dictionary is not being updated by DesignBais (see the descriptions for the System Parameters fields "Routine to Call to Verify Dictionary"). In **Selection Processes**, Fields to Display will output using ITypes if

this box is checked. All ITypes used in this manner must be manually compiled whenever they are updated, whether by DesignBais or by other means.

Failing to compile the I-Type will cause an error:

'**DesignBais Application Error** – An unexpected error has occurred (1)'

If you run DBI.RUN.LAST in a telnet session the error will display:

Program 'DBI.G.SETUPRWNET': Line nnn, I-descriptor must be compiled before execution.

Refer to the section in this Reference Manual "**Creating I-Type Dictionaries**"

Output Justify — Controls the horizontal position of the data within an input or output field. Fields within a multivalue grid take their justification from the field type, so right justification is set for numeric type fields.

Vertical Justify — Controls the vertical position of the data within an input or output field.

Input Conversion Subroutine

If there is a requirement to perform non-standard input conversion, a subroutine may be called. This routine will provide DBVALUE as the entered value. The value returned must be in DBVALUE. PROCESS.EVENT will be set to "INPUT CONVERSION".

Output Conversion Subroutine

If there is a requirement to perform non-standard output conversion, a subroutine may be called. This routine will provide DBVALUE. The value returned must be in DBVALUE. PROCESS.EVENT will be set to "OUTPUT CONVERSION".

Lookup File — Assigning a **Lookup File** (See File Definition) to a field will automatically create a dropdown list for the field every time that field is placed on a form. If the lookup file has more than one selection criteria assigned, you can nominate the required selection by appending ',*Selection Number'* to the lookup file.

Eg. DBCOUNTRY,2

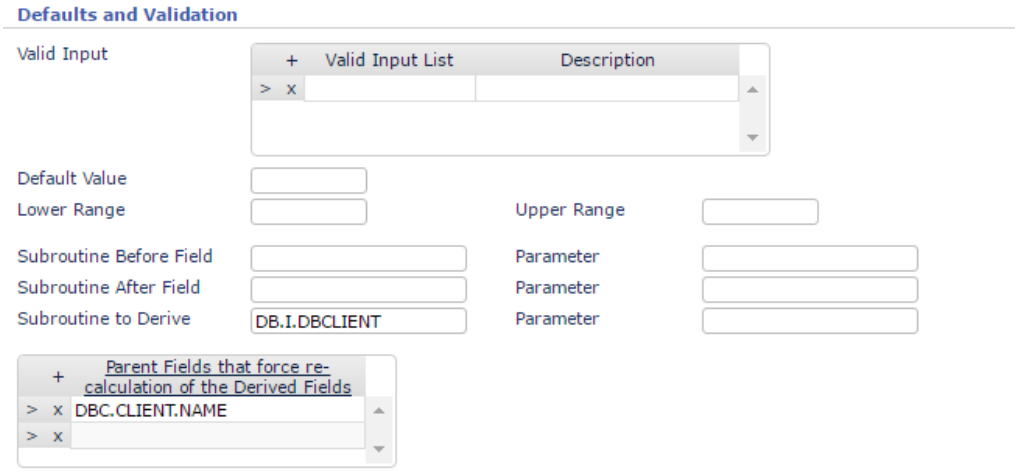
Display in Select Window — If this field is set to "Yes", then the field will automatically be loaded as a display field in search forms associated with the file. This will save a lot of time in the creation of search forms and help to provide a standard interface in search forms.

Help Text — When assigned to a field it is loaded on all forms that contain the field. When the user presses *F1*, the help text associated with the field will be displayed. The Help Text will also be used if you generate DesignBais help for a form.

The use of *F1* to display help for a field relies on the field having focus in the browser. Fields with dropdown selection will trigger the dropdown when they obtain focus and therefore the *F1* key will not display the Help Text. To overcome this it is necessary to place focus on a close-by field and tab (or shift+tab if the field is higher in the tab sequence) to the field with the dropdown. Once the field is focussed then *F1* will trigger the display of the Help Text.

Edit — Click the *Edit* button to edit the help text in a bigger window.

Defaults and Validation

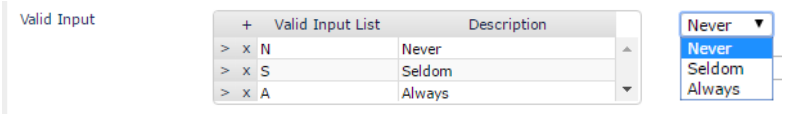


Prompts

Valid Input

Provides for the definition of standard responses for a field. When a field that contains such a list is placed on a form, DesignBais automatically creates a dropdown list. This provides for a very intuitive user interface and reduces server interaction.

The following is an example of a Valid Input list definition and an example of how it is displayed on an entry form. Note that HTML control characters such as '&' must not be used in the List or Description.



Default Value

Establishes a default value for the field. If the default is used in conjunction with a dropdown list, the list item containing the default value will be selected.

Lower Range

Determines the lower range boundary for input values assigned to the field for numeric or date field types only. For a date it may be a valid date or a calulation of the form T, T+nnn or T-nnn where T is the date at run time plus or minus a number of days. Note that this value is overridden by a *min* or *max* "Custom Attribute".

Upper Range

Determines the upper range boundary for input values assign to the field for numeric or date field types only. For a date it may be a valid date or a calulation of the form T, T+nnn or T-nnn where T is the date at run time plus or minus a number of days. Note that this value is overridden by a min or max "Custom Attribute".

Subroutine Before Field

Is used to define a subroutine to be called when the field gets focus on a form. The entry point in the BASIC subroutine is "BEFORE FIELD". This process should be avoided if at all possible since the process will be called every time the field gains focus. Use "C:" to enter a code block. (See section on Code Block Usage.)

Parameter

This field is used to provide extra detail to the subroutine call for the Before Process. The value assigned to this field will fill the variable PROCESS.PARAMETER when the subroutine is called.

Subroutine After Field

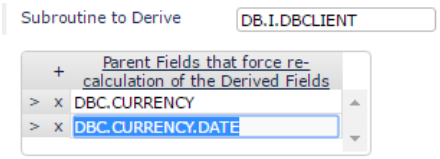
Is used to define a subroutine to be called when the value in the field changes. The entry point in the BASIC subroutine is "VALIDATE". If data does not change when a user changes focus from a field, no event is fired unless "Change Event Will Fire On Loss Of Focus" is set in the Forms Designer. Use "C:" to enter a code block. (See section on Code Block Usage.)

Parameter

This field is used to provide extra detail to the subroutine call for After Process. The value assigned to this field will fill the variable PROCESS.PARAMETER when the subroutine is called.

Subroutine to Derive

Is used to define a subroutine to be called to derive the field's value. When this field is filled with a subroutine name, the following field will be displayed:

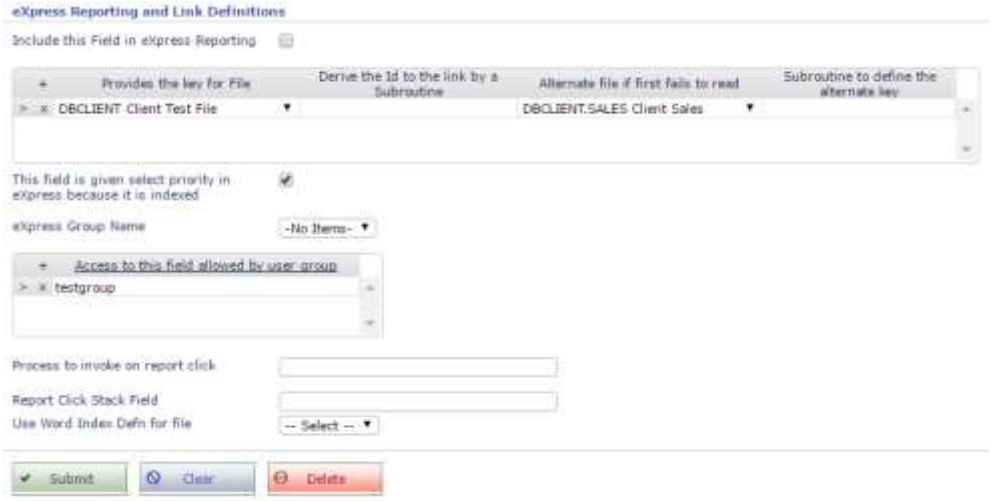


In the above example a field, say DBC.FOREIGN.AMOUNT, will be re-calculated (the defined subroutine DB.I.DBCLIENT will be called) whenever the fields, nominated in the Parent Fields list, change. This provides a very simple method for watching other fields and triggering activity when any field on a form changes. Use "C:" to enter a code block. (See section on Code Block Usage.)

PROCESS.EVENT will be set to "DERIVED". Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information

Parameter

This field is used to provide extra detail to the subroutine call for the Derived Process. The value assigned to this field will fill the variable PROCESS.PARAMETER when the subroutine is called.



Include this Field in eXpress Reporting

This prompt is used to determine whether the current field is to be included in eXpress Reporting.

You may also update this field directly by clicking on the in the “eXpress Reporting” column in the “List all of the field Properties” listing.

Field Name	Screen Label	Attrib	Type	eXpress Reporting	Length	Dec	Just	Conv	Mult	Work	Group
DBC.CLIENT.CODE	Client Code	0	A	✓ Yes	12		L		N	N	
DBC.SESSION.KEY	Session ID	0	N	✓ Yes	10		L		N	N	
DBC.SESSION.ID	Session ID	1	A	✓ Yes	42		L		N	N	
DBC.IDAPP	IDApp	2	A	✓ Yes	42		L		N	N	
DBC.IDBROWSER	IDBrowser	3	A	No	42		L		N	N	
DBC.IDSESSION	IDSession	4	A	No	42		L		N	N	
DBC.IDHIT	IDHit	5	N	No	5		L		N	N	
DBC.IDWINDOW	IDWindow	6	A	No	42		L		N	N	
DBC.HTML	HTML String	7	A	✓ Yes	20		L		N	N	
DBC.HTML.ENCODE	HTML Encode	8	A	✓ Yes	3		L		N	N	
DBC.HTML.MV	HTML Strings	9	A	✓ Yes	200		L		Y	N	HTML
DBC.HTML.ENCODE.MV	Output	10	A	No	50		L		Y	N	HTML

Provides the key for File

This prompt is used to define a file link for the active field. In the example above the active field provides the key-link to the file DBCLIENT.

Derived the Id/Key to the link by a Subroutine

In some cases the link to another file may not be a direct reference to a field. In these cases a subroutine or a dictionary can be used to calculate the direct reference to the key. If the reference is via a subroutine, then enter the subroutine name in this field. The subroutine must have DBI.COMMON included and return the derived key in DBVALUE.

```
PROCESS.EVENT             is "EXPRESS LINK"
PROCESS.EVENTSOURCE       is the Name of the field
PROCESS.PARAMETER         is NULL
DBMVCOUNT                 is set to the multi-value counter if the field is multi-valued

SUBROUTINE GET.KEY
$INCLUDE DBI.COMMON
*
BEGIN CASE
   CASE PROCESS.EVENTSOURCE = "PRODUCT"
      DBVALUE = FIELD(DBRECORD,"*",2)
END CASE
RETURN
```

Alternate file if first fails to read

This is useful if there are history files of archive files that are to be read if the first file fails.

Subroutine to define the alternate key

In some cases the link to the alternate file may not be a direct reference to a field. In these cases a subroutine or a dictionary can be used to calculate the direct reference to the key. If the reference is via a subroutine, then enter the subroutine name in this field. The subroutine must have DBI.COMMON included and return the derived key in DBVALUE.

eXpress Group Name

If there are eXpress Groups nominated for a file, a group may be selected to group the field that is currently being defined. If no group is defined for a field, it is added to a miscellaneous group.

Access to this field allow by User Group

You may nominate a User Group for a Field. This enables you to allow or restrict access to fields within the eXpress report writer depending on the group or groups that a user may belong to.
This option is not enabled for key fields.

Process to invoke on report click

This field is used to nominate a program or form to invoke when this field appears on a report. The field will be clickable on a report. When the user clicks on the field it will invoke the nominated program.

PROCESS.EVENT = “REPORT CLICK”
DBVALUE = The value in the report

Report Click Stack Field

If the previous field has a form nominated, you may define the field to pass the value to.

Use Word Index Defn for file

Specify the Word Index Definition file that is to be used when selecting records based on this field.

Buttons

Submit

Will update the field property record on the DBIPROP file.
See the descriptions for the System Parameters fields “Routine to Call to Verify Dictionary” and “Always Update Base Dictionary”.

Clear

Will clear the form and not save any changes made on the form.

Delete

Will remove the field properties record from the DBIPROP file.

Great care should be taken when deleting a field. Any form that contains the deleted field will be inoperable. Live dictionary items may be updated unless otherwise directed.
See the descriptions for the System Parameters fields “Routine to Call to Verify Dictionary” and “Always Update Base Dictionary”.

Chapter 4 – Menu Definition

Menu Definition

There are currently two types of menus available within DesignBais.

Records updated in this form update the file **DBIMENUS.**

Top Menus

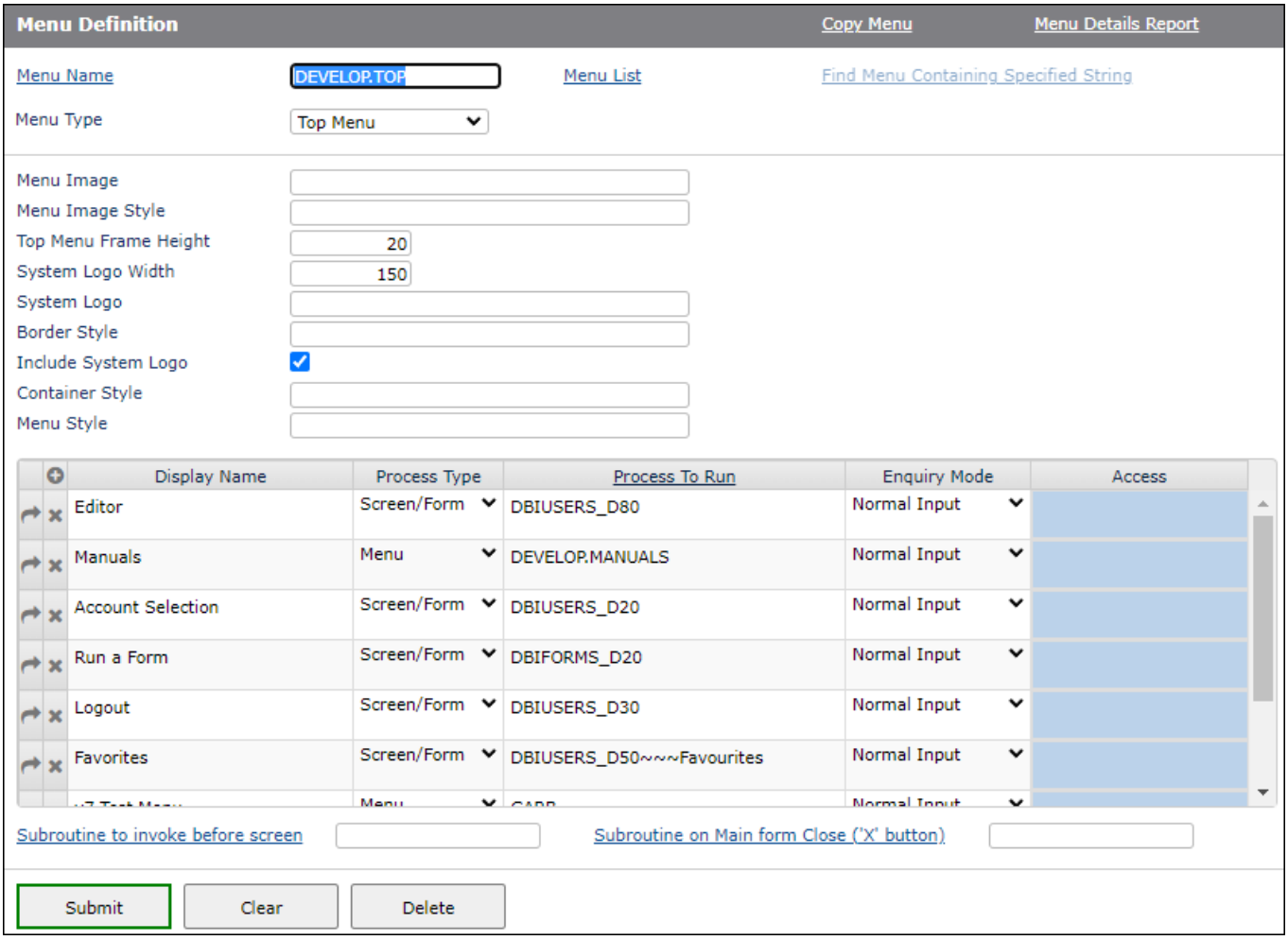
Top Menus can be single level in nature. Alternatively, they can be invoked from other menus to form the dropdown menu structures that most browser users are familiar with. All explorer-style sub-menus must be a top menu style. The DesignBais top menu can be customised. Refer to the self-documenting file in the Web Components css folder called **compatible_w3c.css.**

Side Menus

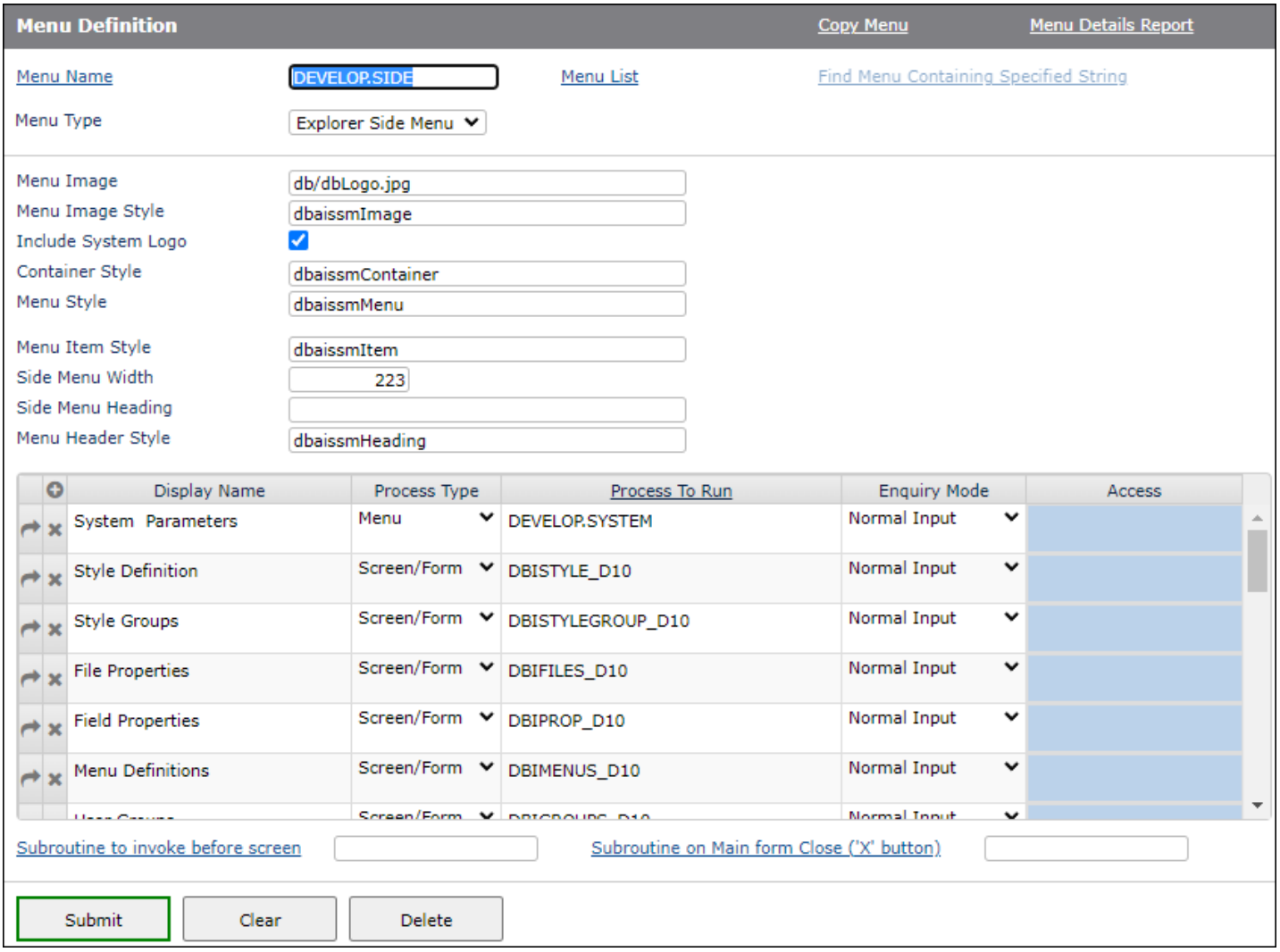
Side Menus can be used to display a small number (usually a maximum of about 15) of options that are more targeted to the end-users' functions. Side Menus can also be hidden altogether.

Side Menus can also be displayed in an Explorer style. This provides an unlimited depth and width for side menus.

Menus are linked to forms. You have the option of assigning a Top and/or Side menu to every form created. This means that you have complete control over the menu structure, the images associated with each menu and the system logo that appears in the top left of the DesignBais window.



Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3



Prompts

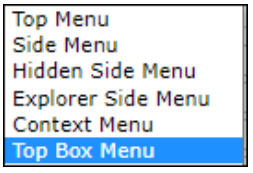
Menu Name

Provides a name for the menu. You should not include characters that are not valid for record ids (keys) in your database platform. Otherwise, names of menus are not restricted.

Click on the Menu Name hyperlink to invoke a search for menus that are defined to DesignBais.

Menu Type

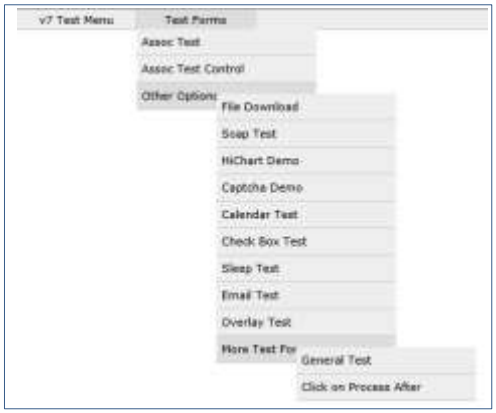
Is used to define the type of menu to create.



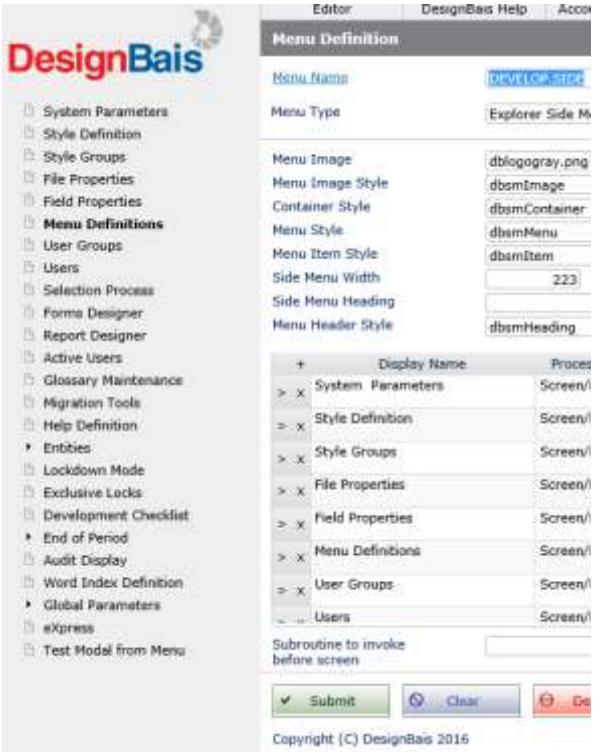
Top Menu

Top Menus can be single level in nature. Alternatively they can be invoked from other menus to form the dropdown menu structures that most browser users are familiar with. All explorer-style sub-menus must be a top menu style.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3



Side Menu — Side Menus can be used to display a small number (usually a maximum of about 15) of options that are more targeted to the end-users functions.



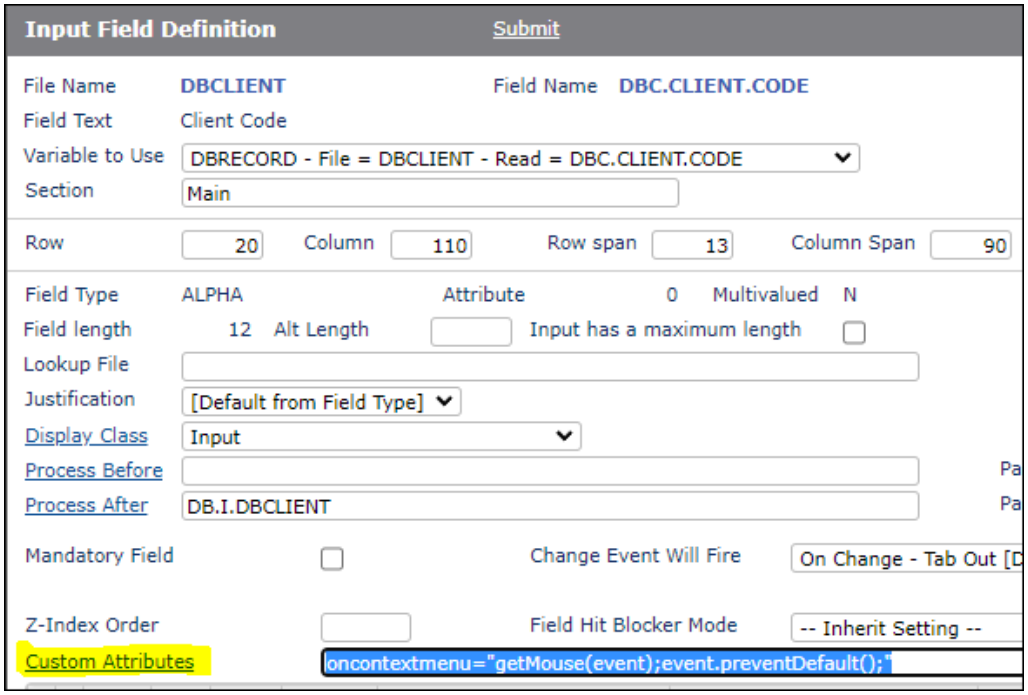
Hidden Side Menu — Will remove the side menu panel altogether. Side menus will not be displayed

Explorer Side Menu — Will display menus in an explorer-style format as shown in the image below.

Context Menu

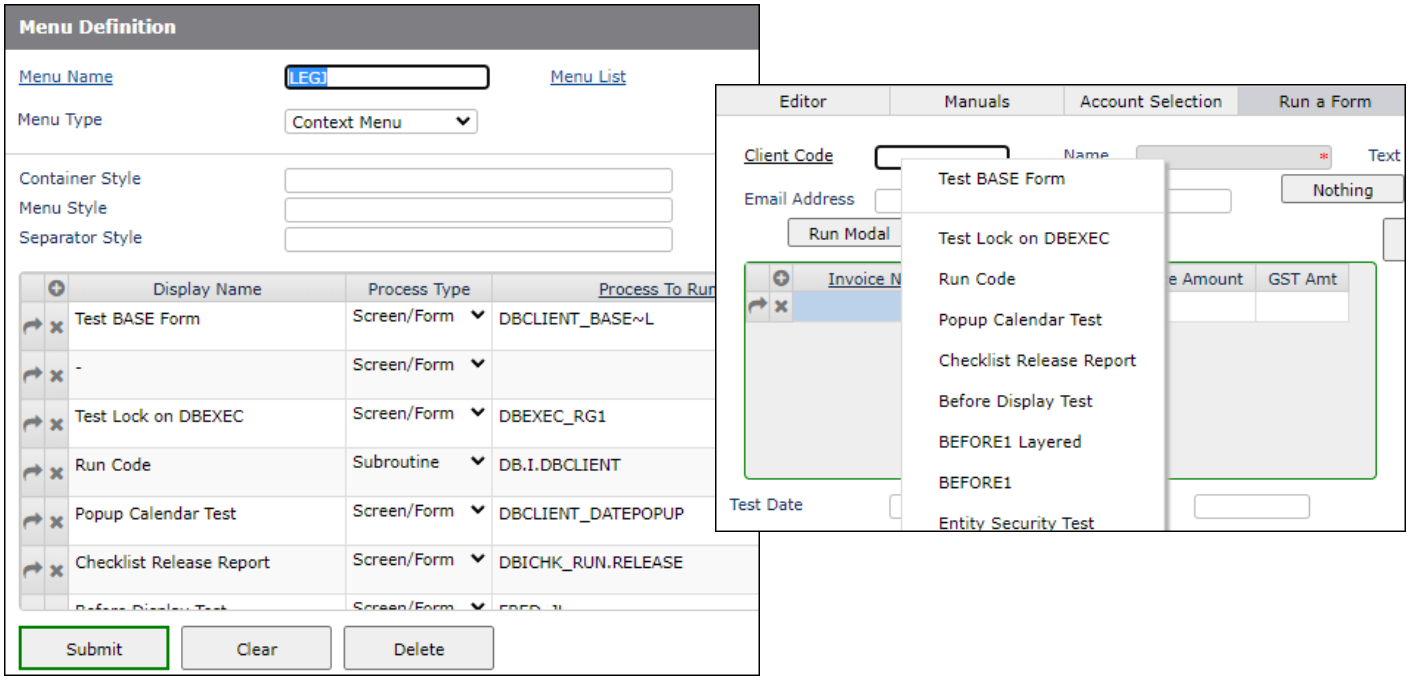
A context menu can be triggered by a mouse right-click within a field when running a form or, in Forms Designer, by a right-click on the canvas. A new event “CONTEXTMENU” has been introduced from Release 8.5.1.6 onwards.

Set the *oncontextmenu* custom attribute on the field. This will cause a right-click on the field at runtime to trigger the *CONTEXTMENU* event.



The basic subroutine event handler defined in the process after slot of the field can then set the menu to be displayed. Sample code is shown below.

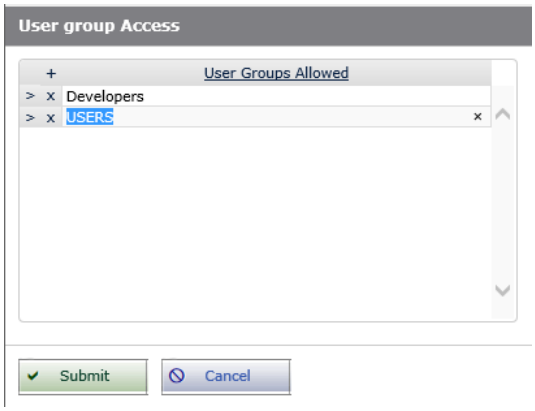
```
CASE THIS.EVENT = "CONTEXTMENU"
        GOSUB CONTEXTMENU

        *
        CONTEXTMENU:
        *
              DBRETURN.TO.FIELD = "NOFOCUS"
              BEGIN CASE
                 CASE EVENTSOURCE = "DBC.CLIENT.CODE"
                    MENU.ID         = "LEGJ"
                    MENU.ITEMS      = ''
                    CONTAINER.STYLE = 'dbaiscmContainer'
                    MENU.STYLE      = 'dbaiscmMenu'
                    SEPARATOR.STYLE = 'dbaiscmSeparator'
                    IF DBWLEVEL = 1 THEN
                       XPOS = 360
                       YPOS = 55
                    END ELSE
                       XPOS = 140
                       YPOS = 35
                    END
                    XPOS=''
                    YPOS=''
                    DWIDTH = 180
                    DHEIGHT = 'auto'
                    CALL DBI.G.CM(MENU.ID,MENU.ITEMS,CONTAINER.STYLE,
                                  MENU.STYLE,SEPARATOR.STYLE,XPOS,YPOS,DWIDTH,DHEIGHT)
              END CASE
              RETURN
```



Top Box Menu Create a box menu that can be attached to the initial top menu rather than to a sub-menu. The top menu will display a containing an icon while the menu itself is a structure. Refer to Top Menu below.

Based on the Menu Type the following fields are displayed:

Menu Image	An image that is to be associated with either a top or side menu. It allows the developer to control the logos that appear in the top left hand corner of a DesignBais form. This provides great control over the look of the interface. It allows the developer to define images that represent different modules of the application.
Menu Image Style	The style to be applied to the Menu Image.
Top Menu Frame Height	Used in top menu definitions only. It controls the height of the topmost DesignBais frame. The developer can remove this frame completely by creating a top-menu with a height of zero. The default is 100 pixels. The Top Menu Frame Height is set to contain the menu plus the height of a “Menu Image”. If you need an image between the Top Menu and the Form you may need to include a margin or padding in the “Menu Image Style” to make the image fully visible. Naturally the Top Menu Frame Height should be increased to accommodate the image.
System Logo Width	Controls the width of the system logo space in the top left hand corner of the form. The default is 150 pixels. Changing this value will also shift the leftmost position of the top menu.
System Logo	The system logo entered here will override the main system logo entered in the system parameters section. The system logo will be displayed in the top left hand frame in the DesignBais window. The image name entered here must reside in the webserver’s image directory. The resolution of the image should be 135 pixels wide and 95 pixels high.
Container Style	The style applied to the top or side menu container. Menu Container styles such as “dbaissmContainer” may be copied to a new name. The new style can be modified by the developer and used on the developer’s menus. **The caveat is that if changes are made and DesignBais does not then display correctly it is the developer’s responsibility to find and fix the problems. Amending positional and dimensional attributes of menus is done at your own risk.** In DesignBais, the side menu class is fixed as "dbaissmContainer". If a developer uses some other CSS class they must make sure that the positional and dimensional attributes are not affected by copying from dbaissmContainer.
Menu Style	The style applied to the top or side menu.
Menu Item Style	The style applied to the item description within a menu.
Side Menu Width	Used in side menu definitions only. This value controls the width of the side menu. If the width is not set then the default of 150 pixels is used.
Side Menu Heading	Used in side menu definitions only. This is the text heading that will appear at the top of the sidebar menu.
Side Menu Heading	The style to be applied to the header text within a menu.
Access	The access level indicates the user groups that are allowed access to the menu option. If you leave this field blank then all users are provided access.
	When you click in the field it will invoke the access level form as shown below:



You can assign the User Groups that are allowed access to the menu option. Clicking on the User Groups Allowed hyperlink will invoke a search form that displays all available user groups.

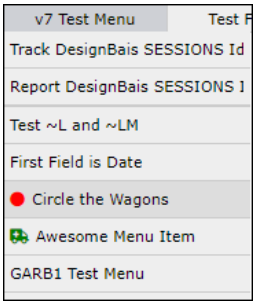
Display Name

The text displayed on the menu.

It is possible to place an image here instead of text:

<img alt="Home" src="images/db/drawer.jpg" style="height:40px;">

The text “Home” will be displayed if the image cannot be located in the website images folder.



The Display Name can also be an awesome font such as:

~fa|fa|fa-circle|red|1x||Circle the Wagons|5||||||fa-solid
~fa|fa|fa-ambulance|green|1x||Awesome Menu Item|5||1||||||Triage

The awesome font string is maintained via a DesignBais form which is invoked by clicking the *Display Name* header after highlighting the row to be maintained. Alternatively right-click the selected row in the Display Name column and use the context menu to open the Awesome Font maintenance form.

Process Type

Determines what type of process to run.

Screen/Form	Will invoke a DesignBais form or a URL. In the above example the first menu option will invoke the DesignBais editor.
Modal Form	Will invoke a DesignBais form and run it as a modal form. This should only be used on side menus.
Menu	For Top Menu structures you can add a menu name in the Process to Run. DesignBais will layer the menus that appear within menu definitions. **Warning – adding a menu to its own menu tree will cause problems within DesignBais as it attempts to resolve the menu structure to display for the form. If the menu is the system start form menu then DesignBais may stop responding**.
Subroutine	Enter a subroutine name in the Process To Run. Use the tilde syntax to set PROCESS.PARAMETER. For example DB.I.SUBR~~~PARAM1 will call the subroutine DB.I.SUBR with PROCESS.PARAMETER set to the value PARAM1.
Box Menu	Not available in this release. See below.

Process to Run

Defines the process to be invoked when the menu option is selected by the user.

Valid Options

DesignBais form name in the format *filename_formname.* Even though DesignBais stores its forms internally as *Filename*formname* you are required to enter them with the '_'. This is to get around some databases' handling of the '*' within directory files.

It is also possible to include run-time parameters to the form like

- *Filename_Formname*~L~E
- A URL. For example http://www.google.com. You must include either http or https in the URL.

If focus is on a grid row that references another menu then clicking the Process to Run hyperlink allows the referenced menu to be maintained or viewed in a layered Menu Definition form. On closing the layered Menu Definition window focus returns to the original menu record.

Enquiry Mode

This option allows you to run a DesignBais form in either Normal Input or Enquiry mode. This feature will ultimately reduce development effort as every maintenance form can be automatically be switched to an enquiry mode form. Only the fields required to input the keys are left enabled. See the Enquiry menu option in the Forms Designer. See entries for ~E in the index for related information.

Subroutine to invoke before screen

The subroutine named in this field will be called whenever a menu option is clicked before the form/screen is loaded. This subroutine can be used to perform any housekeeping tasks required by your application. Sub menus will inherit the subroutine from parent menus if no subroutine name is defined.

```
PROCESS.EVENTSOURCE     = SCREENROOT - the form being loaded from the menu
PROCESS.EVENT           = "BEFORE SCREEN"
```

If menus contain different subroutines then a new subroutine will only take effect in certain circumstances. The active subroutine is held in common variable DBSUBROUTINETOCALL. The subroutine named on the top menu of the first form loaded will be invoked for all forms until a different top menu is invoked, or a side menu with a subroutine is invoked.

Subroutine on Main form Close ("X" button)

The subroutine named in this field can be used to perform any housekeeping that your application requires when the main form is closed. Sub menus will inherent the subroutine named in parent menus.

The PROCESS.EVENTSOURCE is SCREENROOT and the PROCESS.EVENT is "BEFORE CLOSE"

The subroutine named in this field will be called whenever a menu option is clicked before the form/screen is loaded.

Box Menu

In Global Parameters General Parameter set the *Allow Box* Menus to Yes. This flag is held in the record BOX.MENUS on DBIGLOBAL. This enables a “Top Box Menu” option in the Menu Type dropdown and a “Box Menu” option in the Menu Options list.

Create your box menu styles or use the DesignBais styles. Using a standard naming convention will keep style names that start the same together in Style Definition – makes life simpler in the long run when related css is in the one record.

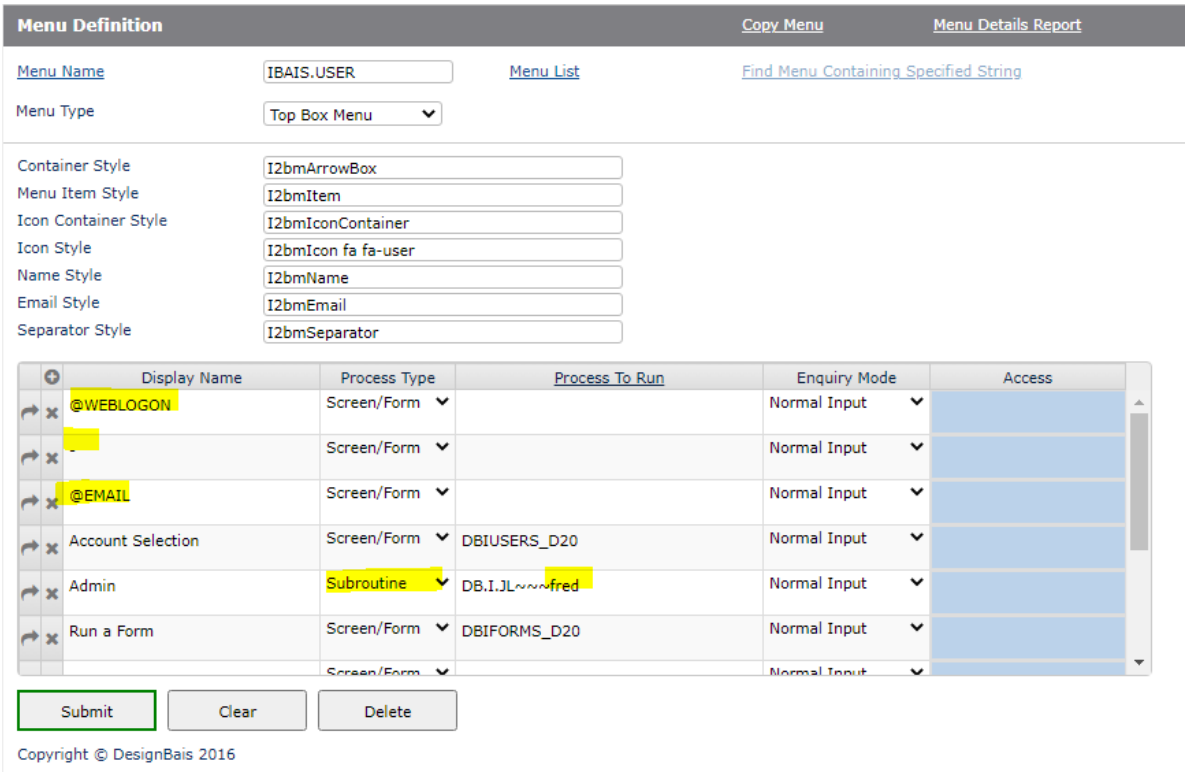
For example , the Style “I2bmArrowBox” has additional styles which you will see as separate items in the attached txt file:

Edit Addittional Styles as Text

```
position: relative
border: 1px solid #ccc
display: none
padding: 30px
position: absolute
left: 1000px
height: 300px
width: 190px
border-radius: 7px
top: 68px
box-shadow: 1px 1px 5px 1px #ddd
z-index: 99999
padding-left: 40px
padding-right: 40px
} .I2bmArrowBox:after, .I2bmArrowBox:before {
bottom: 100%
left: 205px
border: solid transparent
content: " "
height: 0
width: 0
position: absolute
pointer-events: none
} .I2bmArrowBox:after {
border-color: rgba(255, 255, 255, 0)
border-bottom-color: #fff
border-width: 10px
margin-left: -10px
} .I2bmArrowBox:before {
border-color: rgba(204, 204, 204, 0)
border-bottom-color: #ccc
border-width: 11px
margin-left: -11px
```

You may have to adjust the left position, height and other settings to suit your menu system.

Create your “Top Box Menu”. An example follows:

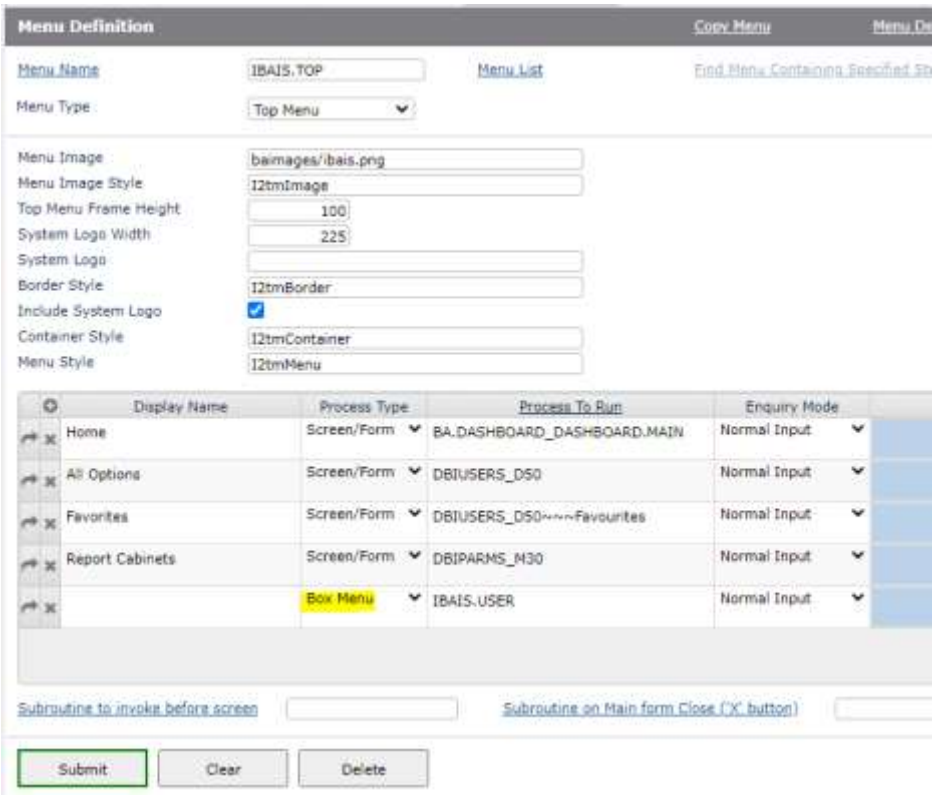


@WEBLOGON displays the logged on user name.
"-" includes a separator line
@EMAIL displays the users email – will invoke an email.

The "Subroutine" option shown will run the subroutine DB.I.JL with the PROCESS.PARAMETER="fred". The value in PROCESS.PARAMETER cab be used to identify which menu option was clicked.

The Awesome Font icon "fa fa-user" to designate the user as shown in the "Icon Style" field setting above. DesignBais is released with FontAwesome 4 but can be upgraded to a later release if required (see data component manual).

Add the Box Menu to your Top Menu:



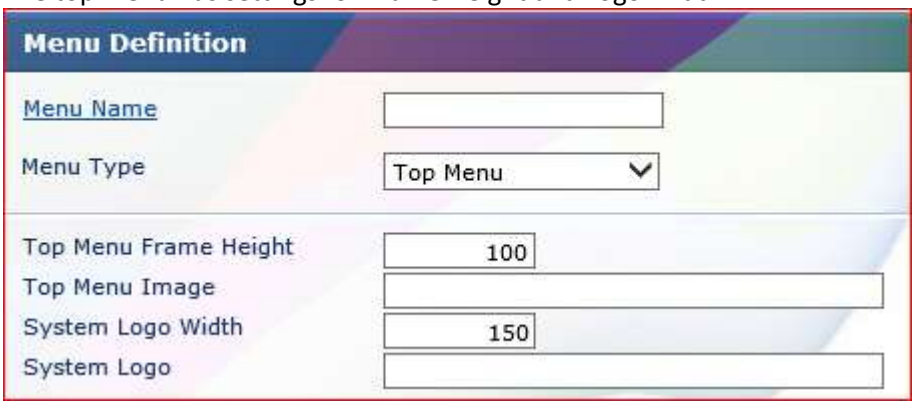
Adjust your top “Menu Style” to push the last menu option to the right:

```
} .I2tmMenu > li:last-child {
float: right
} .I2tmMenu > li:last-child > a {
text-align: right
} .I2tmMenu > li:last-child > a > span {
text-align: center
```

Buttons	
Submit	Updates the file DBIMENUS. The record ID is the menu name.
Clear	Clears the form. Changes are not preserved.
Delete	Deletes the menu from the DBIMENUS file.
Copy Menu	Will invoke a sub-form that will allow you to copy the existing menu.

System Logo

For normal logo placement the image space is determined by the top menu settings.
The top menu has settings for Frame Height and Logo Width:



The top menu frame height can be increased from the DesignBais default of 19 to leave space for an image under the menu. The example below results from setting the height to 130 to match the image height and setting a width of 150.

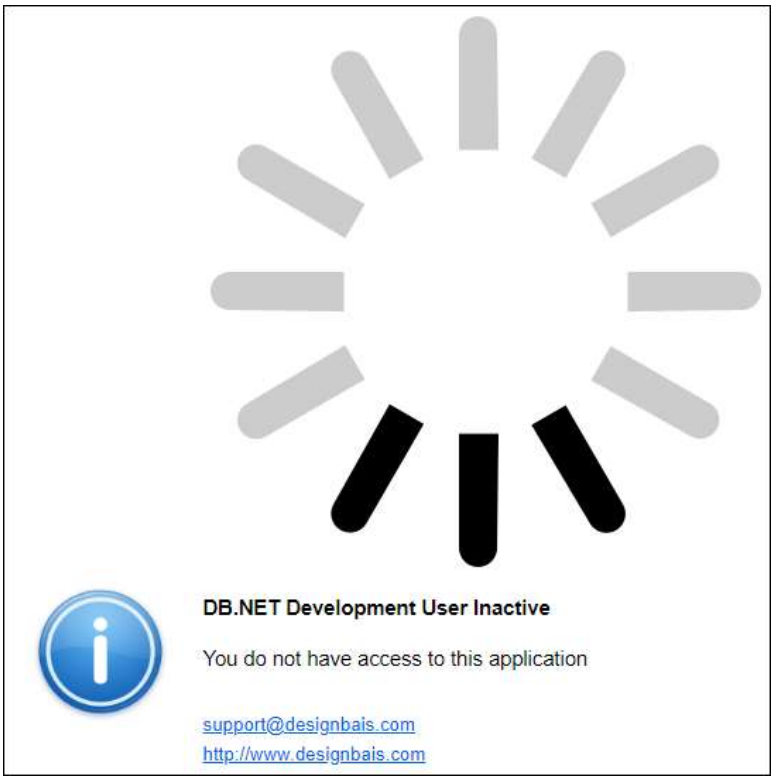
Changing the logo width to 75 demonstrates the sideways distortion of the image.

Vertical distortion occurs if the height is left at 19 (the default value inside the menu height 20).

System Crash

The System Parameter “System Logo” is used on crash errors produced from the database.

Example of system crash when an inactive user attemps to log in to DesignBais:

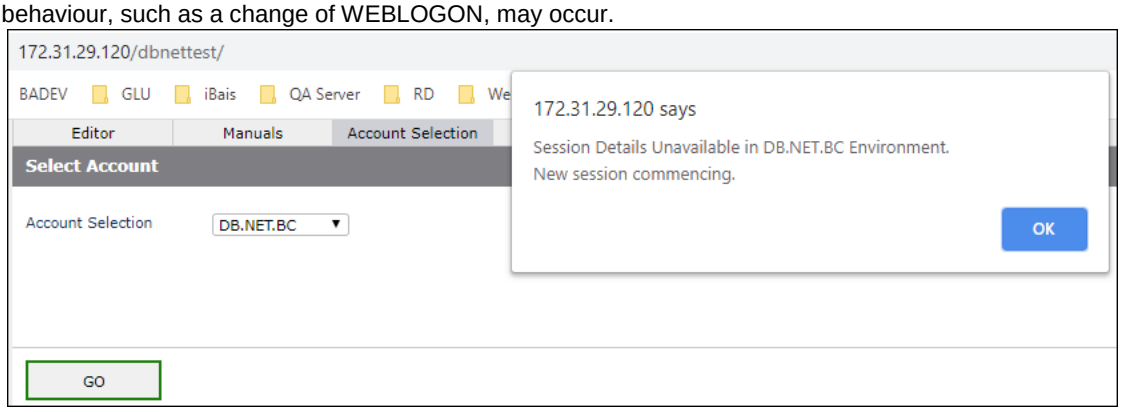


DesignBais Top Menu Options

The name of the menu is DEVELOP.TOP. This menu can be modified as required but the functionality of the standard options listed below should remain available.

Editor	Opens the DesginBais Editor in a new browser tab adjacent to the current DesignBais tab and to the left of any existing Editor tabs that have been opened from the current DesignBais tab. Refer to this manual for help on using the DesginBais Editor.
Manuals	Links to the Web Component and Database Component manuals. usermanual/usermanual.aspx http://www.DesignBais.com/downloads/vDBVERSION/DesignBais Reference Manual.pdf
Account Selection	Runs the form DBIUSERS_D20 which displays a dropdown list of available account names. Select the required account and click the GO button. The DesginBais tab will be directed to the selected account and will display the start form for the user in that account. If the Show Status Bar flag is on (refer to User Maintenance) then the name of the account will be displayed in the bottom left of the form.

Note that DesignBais does not validate the list of Start Accounts and Start Forms on the DBIUSERS record. System Administrators should ensure that all accounts listed in user records, and therefore displayed in the Account Selection dropdown list, use the same DBIUSERS, DBISTATS, DBISESSIONS and DBILICENCE files. It is possible to select an account that uses a different set of these files, perhaps linked to a different website. DesignBais may appear to allow the account to be selected and logged to but unexpected behaviour, such as a change of WEBLOGON, may occur.

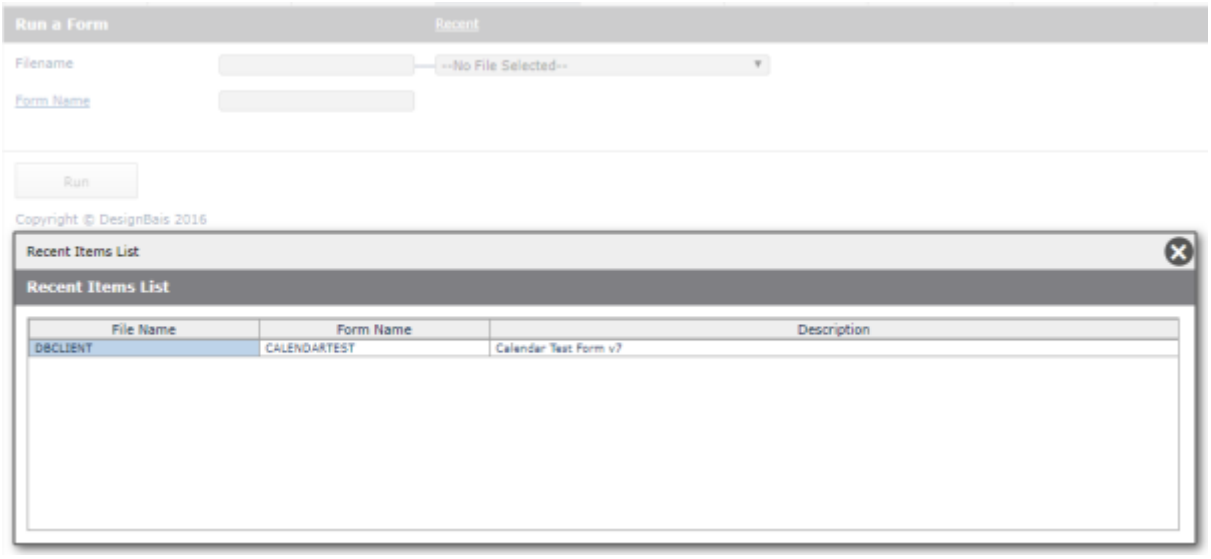


If DBISESSIONS is not shared the above error may display (in this example the account selected from the dropdown is called DB.NET.BC):

Run a Form

Runs the form DBIUSERS_D20 which provides for selection of Filename and Form Name of the form to be displayed. Some users may not know the Filename and Form Name of a required form and the Favorites (Favourites) option is designed to facilitate selecting the form to be run based on a description of the function (see Favorites below).

Click the Recent button to display a list of recent forms that you have run. Click the highlighted column and row to re-run a form.



Logout

Clicking the Logout option on the top menu of any open DesignBais tab, then clicking the Logout button on the form that is displayed, will log the user out of DesignBais and display the message:

“Close all tabs in your browser and/or all browser windows for a more secure log out.”

The tab description will be replaced with “Logged Out”.

Other DesignBais tabs in the browser will remain. If they are activated then the user will be logged in again.

DesignBais keeps a record of each logged in browser and each tab within this browser session.

Closing a browser tab using the “X” in the tab will logout the tab the same as if Logout on the top menu is clicked. Closing a browser and all its open DesignBais tabs using the top right “X” button will logout the user from that browser and properly close all DesignBais tabs.

The list of sessions is not cleared when a new session starts, such as when a user changes account. If the tab close event is received from the web component then only the associated session is deleted from the list. When the list contains no session instances then the record is deleted. A logout will delete the browser record.

The Auto Login option, if turned on in Global or System Parameters, directs DesignBais to check for a session, in the list of sessions for the browser, that has not timed out and pick up the user automatically.

Note, however, that opening a new tab, using *Duplicate Tab* or by entering a url, will force the user to login again if the user has just closed a tab (with X or with the Logout option) and has not activated any other open tab. Note also that Auto login is not applied to URLs containing *dbpage*. This relates to Responsive Design where *dbpage* is used to open a specific form such as in this example:

http://192.168.199.194/dbnet/?ac=ws&dbpage=dbweb-home

DesignBais implements auto login by checking for the user in existing sessions for a browser providing the following conditions are true:

- The user is not opening the code editor
- This is the first hit of a new DesignBais session
- The session has not timed out
- There is no previous logon via IDWindowOld or prevSession
- There is no *dbpage* in the url
- Auto logon is active (System and Global parameters)
- The public user is in the list of “Public Logon Users” or the list of “Public Logon users” is empty (Refer System Parameters)

The system administrator can logout users using the Active Users option on the side menu.

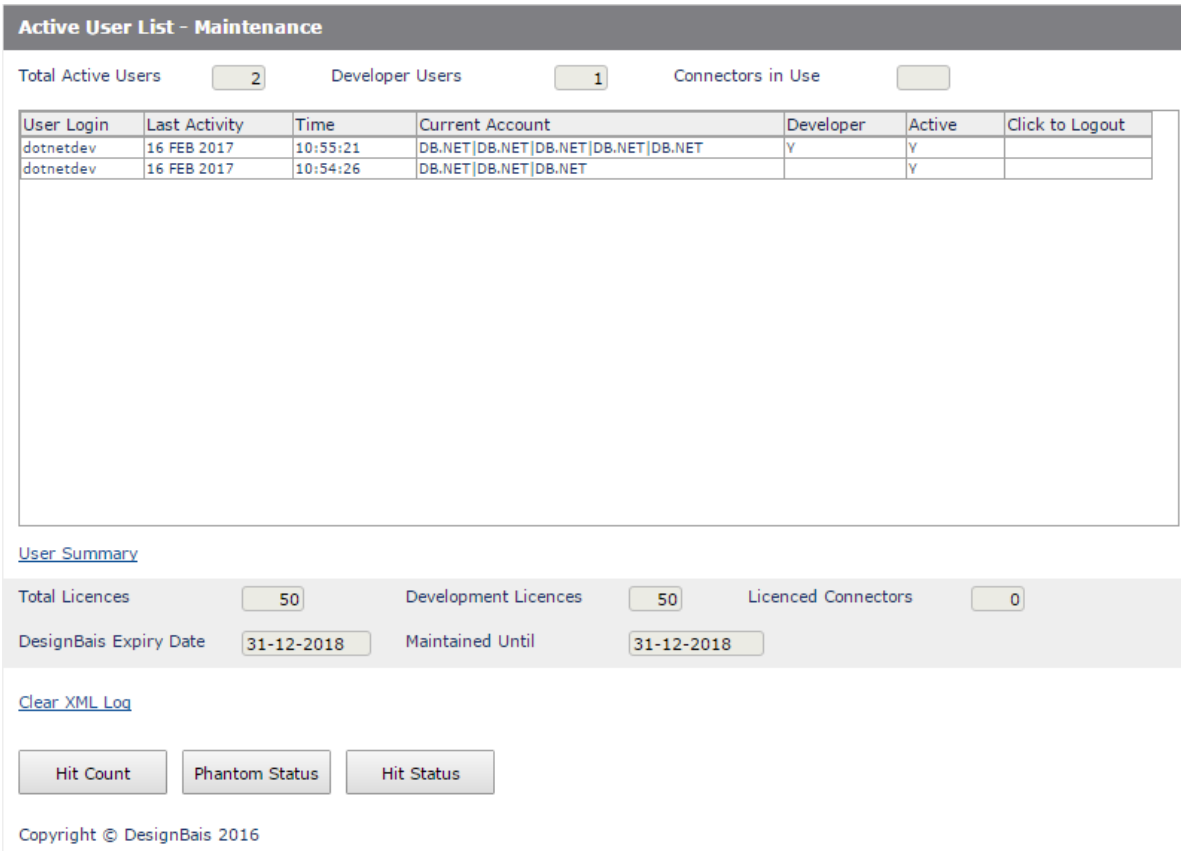
The DesignBais Editor is not effected by the logout of a DesignBais tab. Editor tabs must be closed using the “X” button on the tab.

It is advised that the System Administrator sets up a separate DesignBais user record for the purpose of accessing the Active Users form in the event that the count of active users exceeds the number of licenced users. This user should have a start form of DBIPARMS_D20. The Systems Administrator can then logout one or more User Logins.

To allow the separate DesignBais user to login it is necessary to either have Windows Authenication enabled (and a Windows user id that matches the DesignBais user id) or to set up a qcode configuration for this user id in the db.config file. Refer to the Web Component manual.

Note that normally it is not possible to logout your own User Login in the Active Users Maintenance form but if the start form of your user id is the Active Users form (DBIPARMS_D20) then DesignBais allows this user to log itself out.

If you click ‘Click to Logout’ for your own user login on the Active Users form then you will be presented with the Logout form (DBIUSER_D30). Click ‘Logout’ to log your user out of DesignBais.



If a user ID is in the list of Public Login users (users that start in a login form where the operator switches to their own application user ID) then, when a new tab is opened by that user in a browser, the Auto Login feature will automatically switch to the currently connected user.

If the user ID is not in this list then it is possible to use the db.config file entryPoint node *(?ac=aaaa* - see Web Component Manual extract below) to define a user ID and therefore open adjacent browser tabs with different user IDs.

Note that the Developer flag (Y) is set when a user accesses either Forms or Report Designer. It remains set until the user exits Designer. The user must be in an account that shares DBISESSIONS

6.4.3 entryPoint nodes

The entryPoint node defines the parameters that the WEBCOMP uses to connect to the database. There can be several entryPoint nodes in db.config.

A url like **http://localhost/dbnet?ac=myaccount1** is directed to the entryPoint having "myaccount1" in its qcode attribute.

Set **qcode=""** for the default entryPoint. A url like **http://localhost/dbnet** without the **?ac=xxxx** parameter selects the default entryPoint.

Parameter	Allowed Values	Description
qcode	Attribute	DesignBais sends the request to the entry point having a qcode that matches the 'ac' parameter of the query string.

Favorites

Displays a list of Available Forms and Reports. Click the Recent button to display a list of options that have been run from the Favorites list.

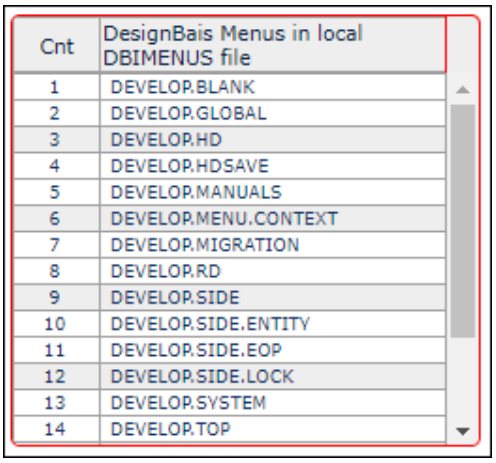
DesignBais System Menus

Menus used by the Designbais tools are released via the DBISYSFORMS file. Menu ids are in the form “MENU*”:menu_name.

For example the DEVELOP.TOP menu is released in DBISYSFORMS with the id “MENU*DEVELOP.TOP”.

Developers cannot use the Menu Definitions form to maintain DesignBais System menus. If you wish to use a Designbais menu in your application then it is necessary to copy it to your local DBIMENUS file, using the *Copy Menu* button, and assign it a different name.

Menus that share the name of a DesignBais system menu, in the local DBIMENUS file, are listed in the report with a red border when you call up the Menu Definitions form. This is to alert you to their presence in your local file. These menus will be deleted if you upgrade to a new DesignBais release.



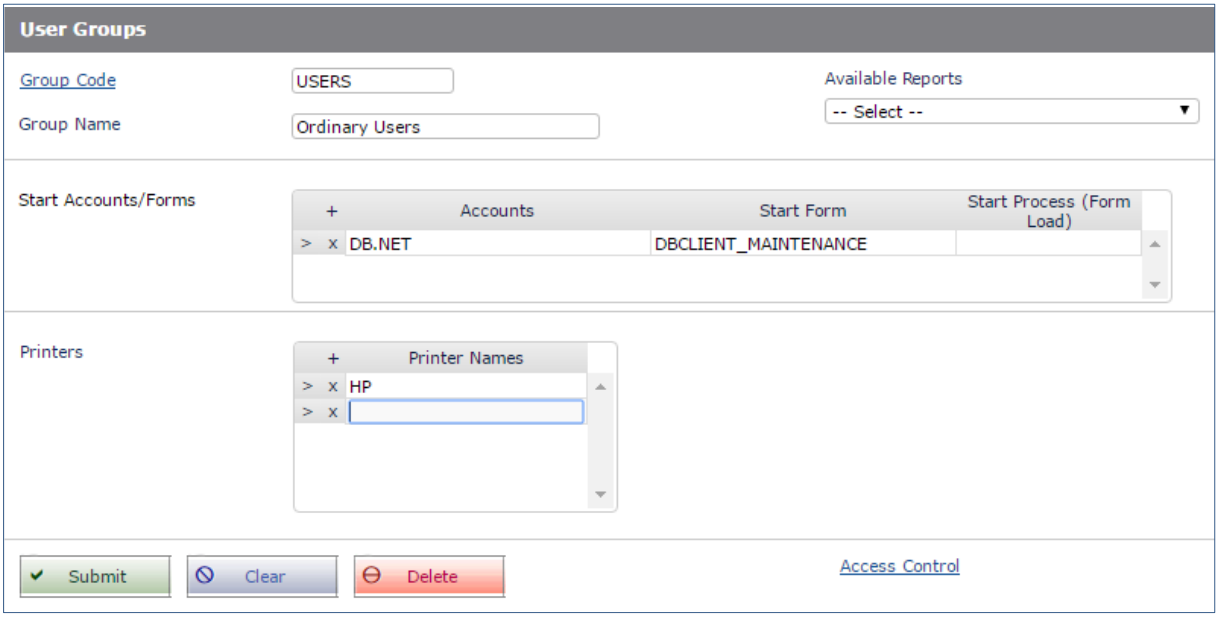
Cnt	DesignBais Menus in local DBIMENUS file
1	DEVELOP.BLANK
2	DEVELOP.GLOBAL
3	DEVELOP.HD
4	DEVELOP.HDSAVE
5	DEVELOP.MANUALS
6	DEVELOP.MENU.CONTEXT
7	DEVELOP.MIGRATION
8	DEVELOP.RD
9	DEVELOP.SIDE
10	DEVELOP.SIDE.ENTITY
11	DEVELOP.SIDE.EOP
12	DEVELOP.SIDE.LOCK
13	DEVELOP.SYSTEM
14	DEVELOP.TOP

Chapter 5 – User Groups

User Groups

User groups provide a level of user security that allows you to group users of a common nature into named groups.

Access to menu levels is controlled by the User Group.

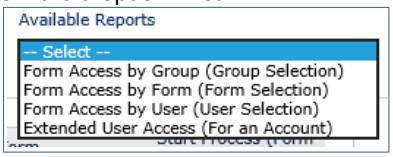


Prompts

Group Code — The name given to a group of users. The group names given to users should be meaningful as this helps to identify them at a later stage.

Group Name — Is used to help identify the group during menu definition.

Available Reports — Select from the dropdown list:



Accounts — Identifies the start account for the user group. This account can be either a Directory or an Account name depending on what is required by the underlying database.

A user group can be given access to multiple accounts. The **Account Selection** option **DBIUSERS_D20** should be added to a user's menu structure, if you are intending to provide access to multiple accounts.

Note that DesignBais does not validate the list of Start Accounts and Start Forms on the DBIUSERS record. System Administrators should ensure that all accounts listed in user records, and therefore displayed in the Account Selection dropdown list, use the same DBIUSERS, DBISTATS, DBISESSIONS and DBILICENCE files. It is possible to select an account that uses a different set of these files, perhaps linked to a different website. DesignBais may appear to allow the account to be selected and logged to but unexpected behaviour, such as a change of WEBLOGON, may occur.

Start Form — Is the DesignBais form that the user will first be presented with. The form can be a blank form that simply contains a menu structure. Alternatively you can direct users to a single form on login that has no menu options. A menu that is contained within its own menu tree will cause problems within DesignBais as it

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

attempts to resolve the menu structure to display for the form. If the menu is on the start form menu then DesignBais may stop responding.

Start Process (Form Load)

Provides an option to run a subroutine every time a form is invoked in DesignBais. This provides the developer the ability to add extra levels of security particular to their application. Alternatively the developer can choose to replace the form that is linked to a menu for another depending on the user or user group.

The event type or entry point for this event is:

PROCESS.EVENT = "GET SCREEN"
PROCESS.EVENTSOURCE = WEBLOGON
PROCESS.PARAMETER = USER.GROUP (the multivalued list of User Groups)

Printers

DesignBais provides the ability to define standard Windows printers for a User Group. The name of the printer should match the name of the printers in the Printer Settings menu within windows.

Buttons

Submit

Updates the file **DBIGROUPS** with the user group record.

Clear

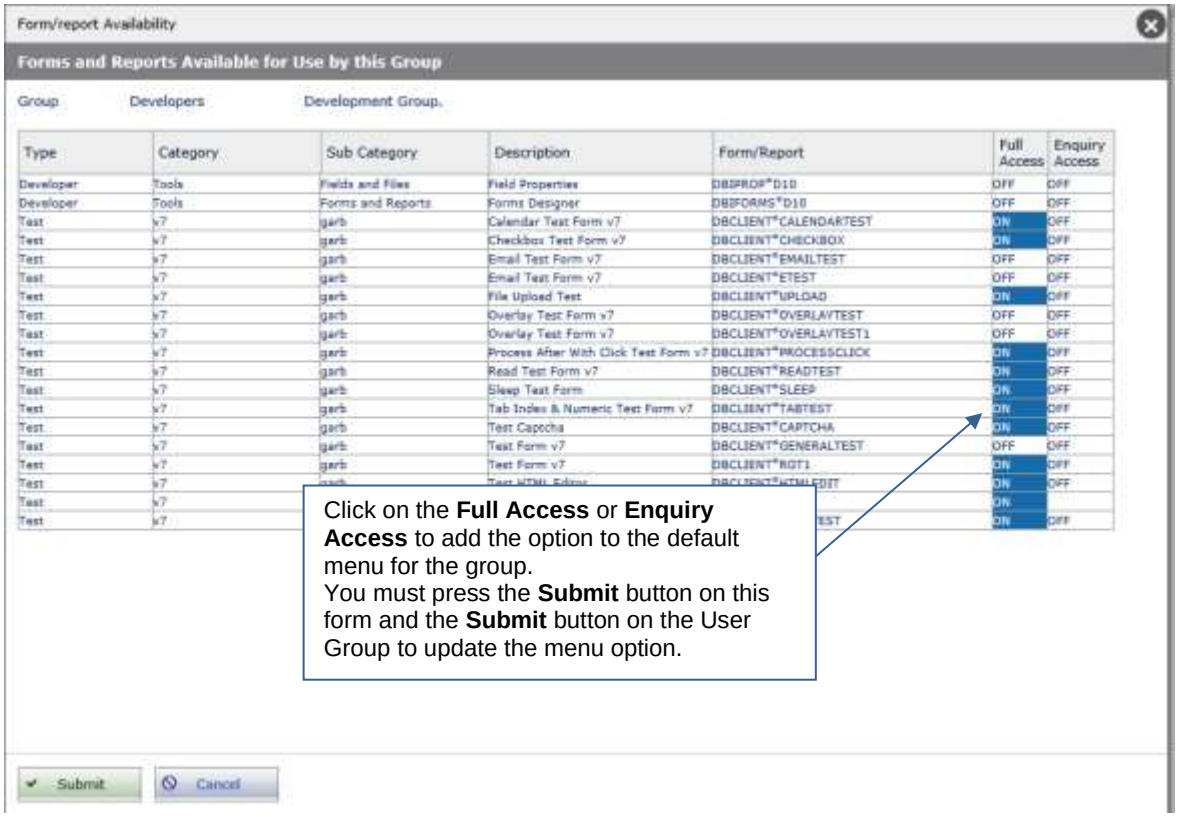
Clears the form and does not preserve changes.

Delete

Removes the current record from the **DBIGROUPS** file.

Access Control

Provides access to the alternate menu structure for the currently nominated group. The form that is displayed will allow you to provide access to the forms and reports available to the group. See the Forms Designer section of this manual for details about setting Type, Category and Sub Category .



Chapter 6 – User Maintenance

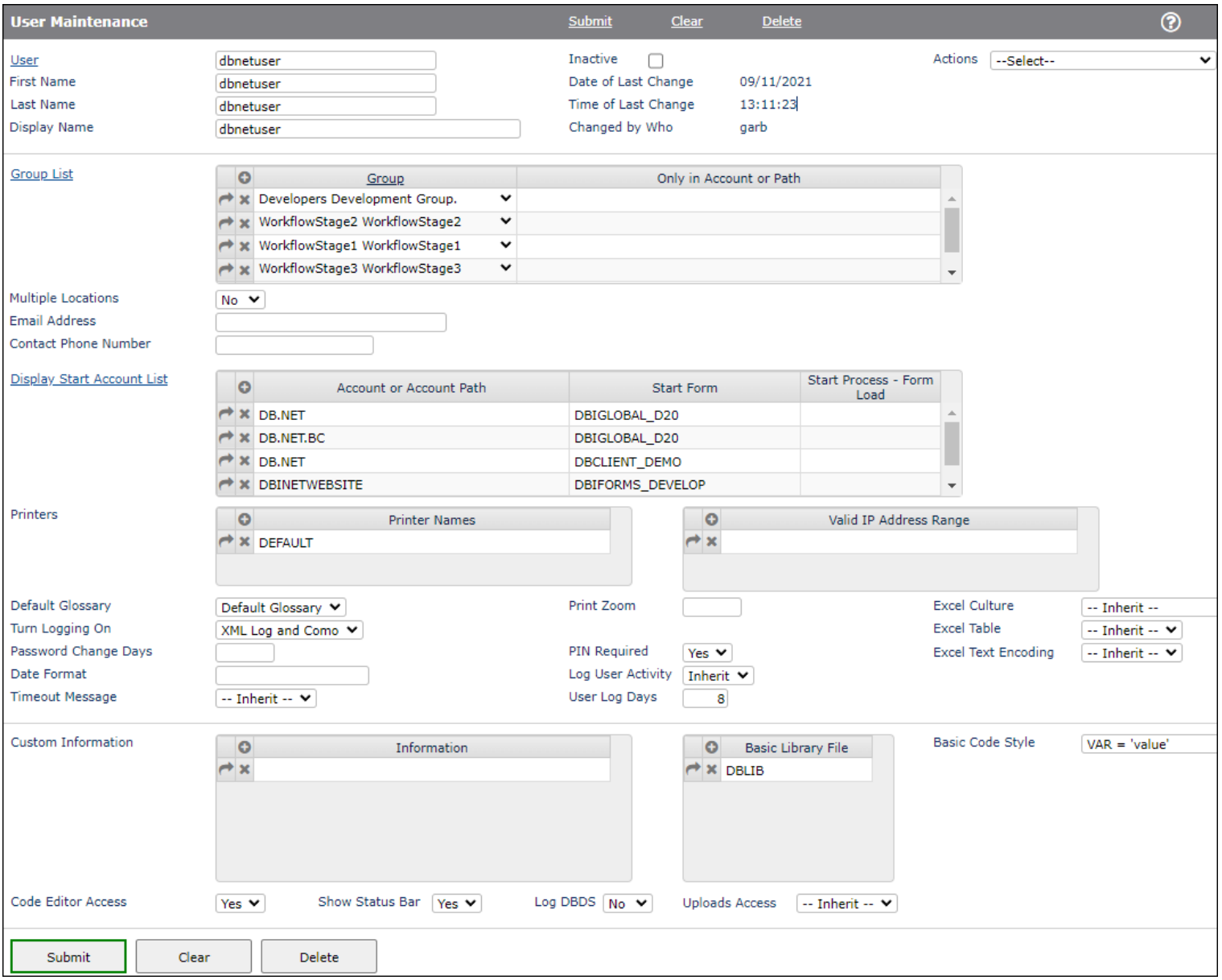
User Maintenance

To gain access to DesignBais the details of the user must be entered into this form.

The user must also be a local user on the Web Server or a user within a Domain that is known to the Web Server

Access to menu levels is controlled by the user record. A user menu level access overwrites group level security.

Section 1:



Prompts

User — The User is a unique name that identifies the user to DesignBais. The user record controls start account, start form, default printers and default glossary for the user. The user must also be a local user on the Web Server or a user within a Domain that is known to the Web Server.

Click on the User hyperlink to invoke a search for users that are defined to DesignBais.

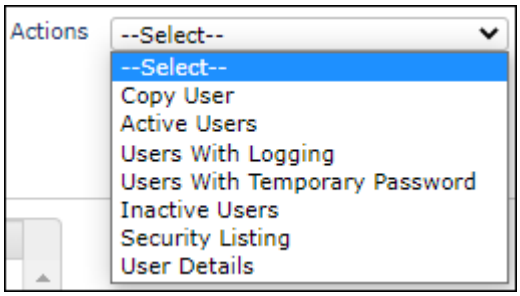
Inactive — Check this box to flag the user as inactive. The user will be prevented from logging into the DesignBais application.

First Name — The first name of the user.

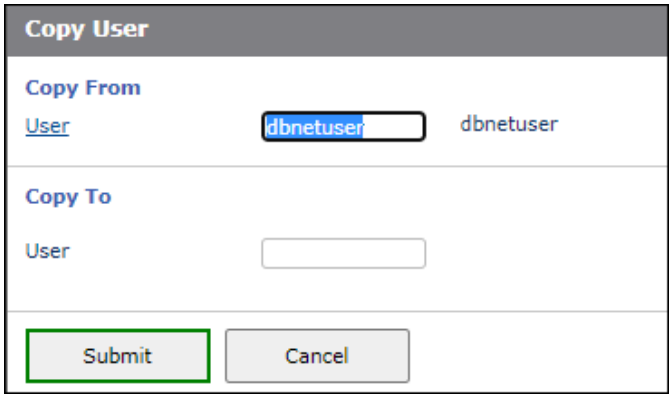
Last Name — The last name of the user.

Display Name — This will be displayed where a user name is required.

Actions — A list of available actions.



Copy User — A new user record can be created based on an existing user record.



The following fields are cleared in the new user record:

- Password Change Days
- Date of Last Password Change
- Password
- Password History

The Date and Time of Last Change to the user record are set. The Changed by Who is set to the WEBLOGON of the user who does the copy.

Active Users — A list of users with the Inactive flag not set. This is the normal setting.

Users With Logging

Displays a list of users with Logging turned on. Logging should only be turned on when required such as to track down a problem. The XML Logging option writes all web to data component traffic to The DBIXMLLOG file which adds a load to the server. The Como option updates the log of user activity on the designated database file, such as &COMO& on a Universe database.

Users With Temporary Password

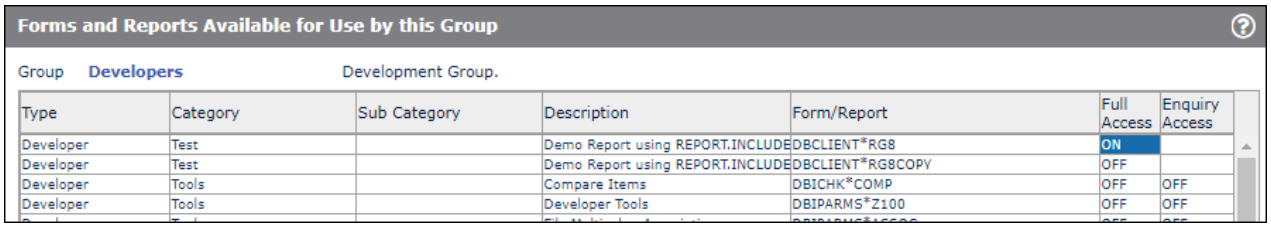
A list of users with a temporary password. When a user requires a password change (either to set a new password or because they have forgotten the password) then a temporary password is emailed to the email address on the user record, provided the user can supply this email address. Temporary passwords are valid for two hours from the time of issue.

Inactive Users

A list of users with the inactive flag set. This flag can be set by the Systems Administrator or when a user fails to login within the number defined by the Failed Login Attempts Allowed in Global User Default Login Parameters.

Security Listing

This report displays the list of database accounts to which the user has access. It also lists the User Groups to which the user belongs and the access available to each group. The access details are extracted from the same source as is displayed by clicking the *Access Control* button on the User Groups maintenance form.



User Details

This option is only accessible to users in the DBAdministrator User Group. It displays a number of non-maintainable user record fields.

Group

Group is used to allocate the user to a user group. Start Account, Start form and printers are inherited from the User Group record if these values are not entered in the user form. A user can belong to multiple groups.

Only in Account or Path

In some cases, you may wish to set a user to a group within a single account. If this is true, enter the name/path of the account in this field. If this field is blank for a group, the user will be assigned that group for all accounts.

Multiple Locations

Assigning "Yes" to the Multiple locations prompt indicates that this user name can be used on more than one computer at the same time. If "No" the user can only be logged in from the one location (device) at any one time.

Email Address

The email address of the user. This is used as a printer device to send reports as Spreadsheets or as HTML via Email.

Contact Phone Number

The contact number for the user. (Informational only, not used anywhere at this time.)

Accounts

Identifies the start account for the user group. This account can be either a System or Account name. Alternatively it can be the directory path route to the account. For sites using mv.NET, this is the Account Profile name, not the actual DBMS account name, though the two may be the same.

A user can be given access to multiple accounts.

The Account Selection option DBIUSERS_D20 should be added to a user's menu structure, if you are intending to provide access to multiple accounts.

Note that DesignBais does not validate the list of Start Accounts and Start Forms on the DBIUSERS record. System Administrators should ensure that all accounts listed in user records, and therefore displayed in the Account Selection dropdown list, use the same DBIUSERS, DBISTATS, DBISESSIONS and DBILICENCE files. It is possible to select an account that uses a different set of these files, perhaps linked to a different website. DesignBais may appear to allow the account to be selected and logged to but unexpected behaviour, such as a change of WEBLOGON, may occur.

Start Process (Form Load)

Provides an option to run a subroutine every time a form is invoked in DesignBais. This provides the developer the ability to add extra levels of security particular to their application. Alternatively the developer can choose to replace the form that is linked to a menu for another depending on the user or user group.

The event type or entry point for this event is:

PROCESS.EVENT = "GET SCREEN"
PROCESS.EVENTSOURCE = WEBLOGON
PROCESS.PARAMETER = USER.GROUP

Printers

DesignBais provides the ability to define standard Windows printers for a User. The name of the printer should match the name of the printers in the Printer Settings menu within Windows.

Valid IP Address Range(s)

The Valid IP Address Range determines what source IP addresses are valid for this user. If this field is left blank then all IP address ranges are accepted. You can enter a partial IP address or a complete IP address. If a partial address is entered DesignBais will validate the part entered against the same part of the source IP address.

Default Glossary

This will set the default glossary for the user on login. It will set the common variable DBGLOSSARY to the value assigned at login. The DBGLOSSARY<1,1> variable can be re-set to another value at anytime during the DesignBais session.

Print Zoom

This parameter provides a means of setting the size of landscape print pages .The browser applies the default "Enable Shrink to Fit" before landscape pages are rotated for printing. If "Enable Shrink to Fit" is left checked then, before rotation, the page is shrunk to fit the page's long margin to the portrait width. This leaves a small printed report. So the *Print Zoom* factor is applied to make the page larger before it is shrunk.

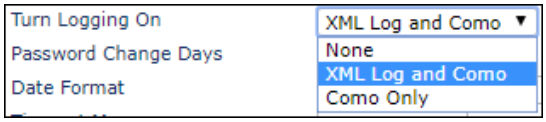
The page needs to be expanded by a percentage enlargement factor of around 160.

If "Enable Shrink to Fit" is switched off by the user than this factor is not required.

If the Global Default zoom is to be ignored then a zero value is required. Leave blank to apply the Global Browser Print Margins default zoom factor.

Turn Logging On

The flag is used to turn logging on for a user. It should not be used as a default for all users as it does incur a performance penalty. When the 'XML Log' option is turned on, all XML transmissions between Client and Database server are logged into the file DBIXMLLOG. The key to the log file is "sequence no":"*":UserName[:"*I" or "*O"]. The 'I' is for input from the Client and 'O' is for output. If the database supports a Como mechanism, and the 'Como' option is turned on then DesignBais will also record all program output in the **como** file. There are two Como records created. UserName, recording the results of the latest transmission and UserName.1 being the history of all transmissions since the latest clearing of the como log file. Refer to "Log File Size" regarding the Como file log.



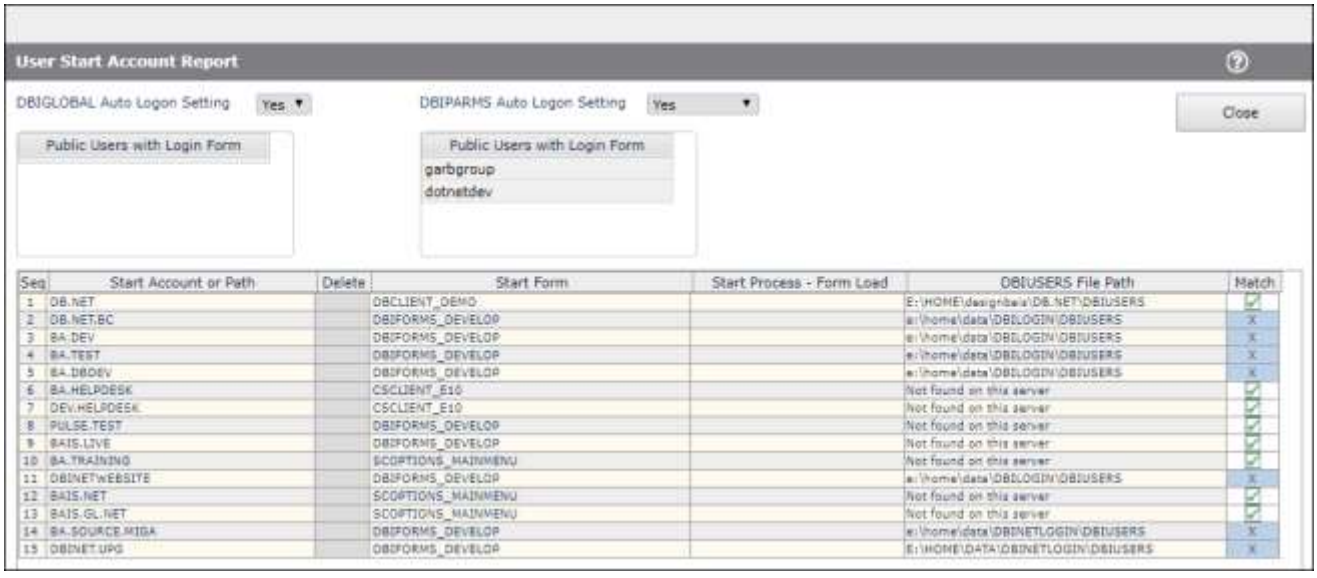
Password Change Days	If you wish to have a policy of regular password changes for a user or user group, enter the number of days in this prompt. When the number of days since the last password change is greater than the number entered in this field, the user will be required to change their password. (This feature is not yet implemented.)
Pin Required	This fields only appears if Two Factor Authentication operating. Users may be exempted from the requirement to enter a Google Two Pass Authorisation PIN by changing this field to No.
Date Format	The date format defined here will override the Date Format in the System and Global Parameters. This field, if populated, is used to ensure that date fields are interpreted correctly by the database server. Refer to the System Parameters Date Format for detailed explanation.
Excel Culture	The Excel Culture is used by the conversion to Excel component to determine if an exported value is a date. The output format is determined by Date Format parameter.The user setting if present, overrides the System Parameter which in turn, overrides the Global setting.
Excel Table	This setting determines if the data exported to Excel is to be displayed in table format and if so, the style of table. If not entered or Inherit is chosen then the setting will be Inherited from the System Parameter which in turn, overrides the Global setting. If No Table Format is selected then this will mean that the raw data is exported to Excel with no table formatting.
Excel Text Encoding	This setting determines if the data exported to Excel is to be encoded as ASCII (default) or as utf-8. Using UTF8 together with the Culture setting allows Currency symbols to be applied when the Excel file is produced. If not entered or Inherit is chosen then the setting will be Inherited from the System Parameter which in turn, overrides the Global setting.
Log User Activity	This User Activity may be logged on the file DBIUSERLOG and reported. This may be set at System or User level as well. Users with Log User Activity set to "Inherit" with use the System Parameters value. Systems with Log User Activity set to "Inherit" will use the Global Parameter. Select "No" to stop logging user activity in the DBIUSERLOG file. "Yes" is the default. Stored in the DBIGLOBAL record USERLOG.
User Log Days	The number of days to keep DBIUSERLOG records.
Timeout Message	When a DesignBais browser tab session has been idle for the Session Timeout period then a timeout message will display if the user attempts to activate the session. The timeout warning message may be suppressed by selecting No. The user will be automatically taken to their start form. A value entered here will overwrite the System Parameters setting.
Custom Information	This provides a multivalue field for storing additional information on the user. The field name is DBIU.INFORMATION and it is stored in attribute 23 of the DBIUSERS file.
Developer Preference	This prompt was used to indicate the preference for the version of the DesignBais forms and report designer. There were two options: a. Pre Version 5 Designer — Utilize pre version 5 Designer. b. Version 5 Designer (IE7+) — Utilize post version 5 designer (Must be on IE7 or above) **This preference is no longer available as at Release 7.1.3.1**
Code Editor Access	Controls access for this user to the DesignBais Code Editor.
Show Status Bar	Controls whether the status bar, at the bottom of the browser window, is displayed. The status bar display is particularly useful for developers as it displays details of the form and field that are in focus in Forms Designer for example, as shown here:

(0) User:dotnetdev - Account:DB.NET - Form:DBCLIENT_EMAILTEST | Field=DBC.CLIENT.NAME | Row=20 | Col=290 | Row Span=18 | Col Span=136 | Section=Main

Log DBDS	Controls the logging of DBDS output which can then be viewed without truncation using the *DBDS Log* button on the Code Editor form.
Uploads Access	If the Secure Uploads feature has been switched on in System or Global Parameters then a user will have access to the website u*ploads* folder only if: • The user login is via a form that utilises DBALTUSER to change the WEBLOGON • The application login form sets SESSION.REC<DBISS.AUTHENTICATED> = "Y" • The user is granted access individually via the *Uploads Access* flag in User Maintenance A user can be denied access individually by setting the *Uploads Access* flag in User Maintenance to *No*.
Basic Library File	The name of the Basic Library files used by this user. The first name in the list will be used as the default in Code Editor. Only applies to users with access to the Code Editor.
Basic Code Style	This setting determines the style of basic code lines that are used in basic subroutines when skeleton or default code snippets are inserted by DesignBais. It defines whether the equal sign in basic code statements is surrounded by space characters and whether single or double quotes are used. Only applies to users with access to the Code Editor.

Buttons

Submit	Updates the file **DBIUSERS** with the user record.
Clear	Clears the form and does not preserve changes.
Delete	Removes the current record from the **DBIUSERS** file.
Copy User	This option will invoke a form that will enable the copying of a user record.
Users with Logging	Displays an On-form Report of all user records that have Logging turned on.
Display Start Account List	This option displays the User Start Account Report which is an On-Form Report of all Start Accounts and Start Forms.



Where a user has a large number of start accounts this report makes it much easier to check if an account is already specified. Click on the column headings to sort the list. Sorting by account name allows you to see duplicates if they exist.

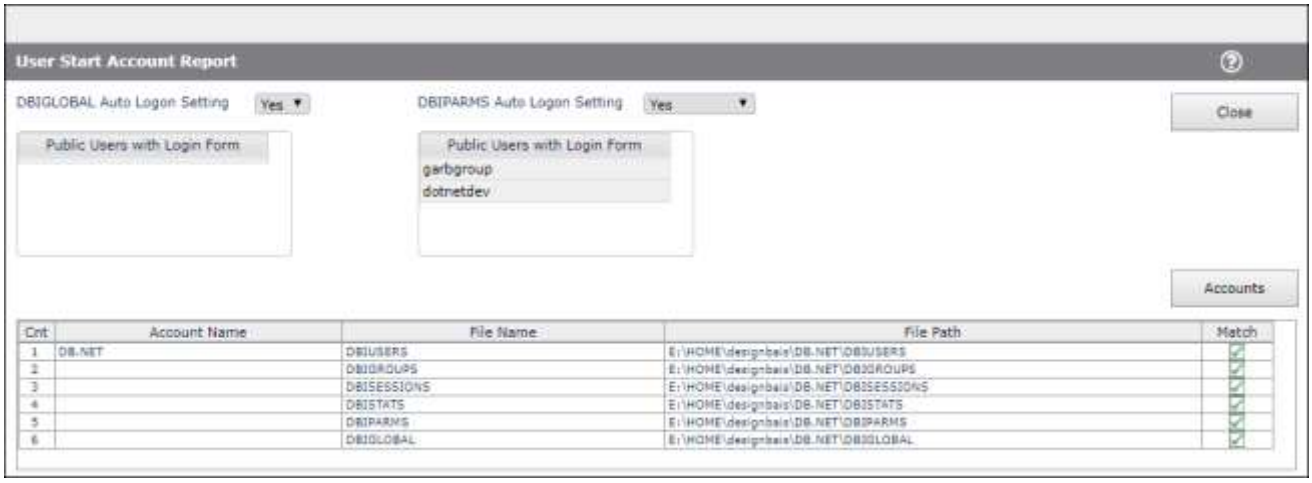
Use the 'Delete' column to flag any rows for deletion. Deletion will occur when the 'Close' button is clicked. Actual deletion is of course only confirmed when the user record is submitted from the main form.

Seq	Start Account or Path	Delete	Start Form	Start Process - Form Load	DBIUSERS File Path	Match
1	DB.NET		DBCLIENT_DEMO		E:\HOME\designbais\DB.NET\DBIUSERS	✓
2	DB.NET.BC		DBIFORMS_DEVELOP		e:\home\data\DBILOGIN\DBIUSERS	X
3	BA.DEV		DBIFORMS_DEVELOP		e:\home\data\DBILOGIN\DBIUSERS	X
4	BA.TEST		DBIFORMS_DEVELOP		e:\home\data\DBILOGIN\DBIUSERS	X
5	BA.DBDEV	Keep	DBIFORMS_DEVELOP		e:\home\data\DBILOGIN\DBIUSERS	X
6	BA.HELPDESK		CSCLIENT_E10		Not found on this server	✓
7	DEV.HELPDESK	Delete	CSCLIENT_E10		Not found on this server	✓
8	PULSE.TEST		DBIFORMS_DEVELOP		Not found on this server	✓
9	BAIS.LIVE		DBIFORMS_DEVELOP		Not found on this server	✓
10	BA.TRAINING	Delete	SCOPTIONS_MAINMENU		Not found on this server	✓
11	DBINETWEBSITE		DBIFORMS_DEVELOP		e:\home\data\DBILOGIN\DBIUSERS	X
12	BAIS.NET		SCOPTIONS_MAINMENU		Not found on this server	✓
13	BAIS.GL.NET		SCOPTIONS_MAINMENU		Not found on this server	✓
14	BA.SOURCE.MIGA		DBIFORMS_DEVELOP		e:\home\data\DBINETLOGIN\DBIUSERS	X
15	DBINET.UPG		DBIFORMS_DEVELOP		E:\HOME\DATA\DBINETLOGIN\DBIUSERS	X

Clicking a column marked 'Delete' will remove the delete flag. The row will display 'Keep' to indicate this. Closing the User Start Account Report form using the 'X' button will cancel all delete flags.

The User Start Account report displays the Global and System Parameters fields *Auto Logon* and *Public Users with Login Form* for reference since these fields relate to the ability of a user to use the auto logon feature. Also relevant to auto logon is the location of some DBI files. Click on the *Match* column to display the details of these 6 files.

The login form provides a method for obtaining the user's credentials in order to open their designated start form. This assumes that the login account and the user's start account share the same DBIGLOBAL file. This would normally be the case, but if not then it could have an effect on, for example, whether a user is prompted for Google Two Factor authentication.



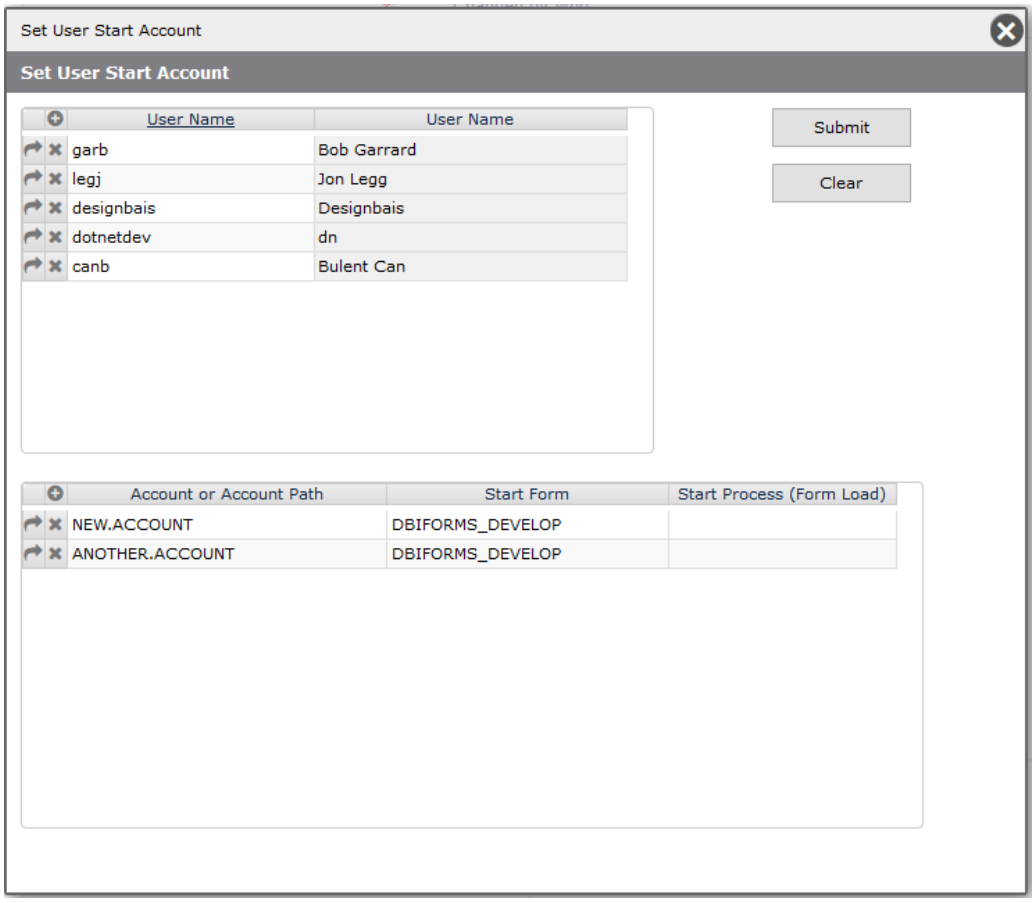
The above example shows that in the DB.NET account all 6 files reside in the same path.

Set Start Account

This option displays a form to allow you to load a set of start account and start form entries to a list of user records. In the following example all the listed user records will be updated.

If the Start Account already exists in the user's list then the Start Form and Start Process will be updated with the values from this form if values have been entered on this form. If the value on this form is null then the current value on the user record will remain.

If the Start Account is not in the user's list then it will only be added if the Start Form has a value on this form.



It is therefore possible to update the Start Process without knowing, or changing, the Start Form. Similarly the Start Account can be updated without knowing, or changing, the Start Process.

User Authentication – Secure Uploads Folder

DesignBais uses the website *uploads* folder for uploading and downloading files to and from the website.

There is a requirement to deny unauthenticated access to the *uploads* folder. To achieve this the DesignBais Data Component flags a session when the user is authenticated, and the web component blocks/allows access to the *uploads* folder accordingly.

How to enable this feature in DesignBais

Note:
This feature should not be enabled when Basic or Windows authentication is the only mode set for the DBNET in IIS. It can be enabled if "Anonymous" or "Anonymous+Basic" authentication is being used.

When IIS is set to Anonymous or Anonymous+Basic Access, authentication can be provided via the DesignBais login form, or a custom login form provided by the application. Note that DesignBais default authentication mode is Anonymous+Basic.

When IIS is set for Anonymous Access, authentication can be provided via the DesignBais login form, or a custom login form provided by the application. In these situations, direct access to static files in *uploads* folder can be controlled as follows:

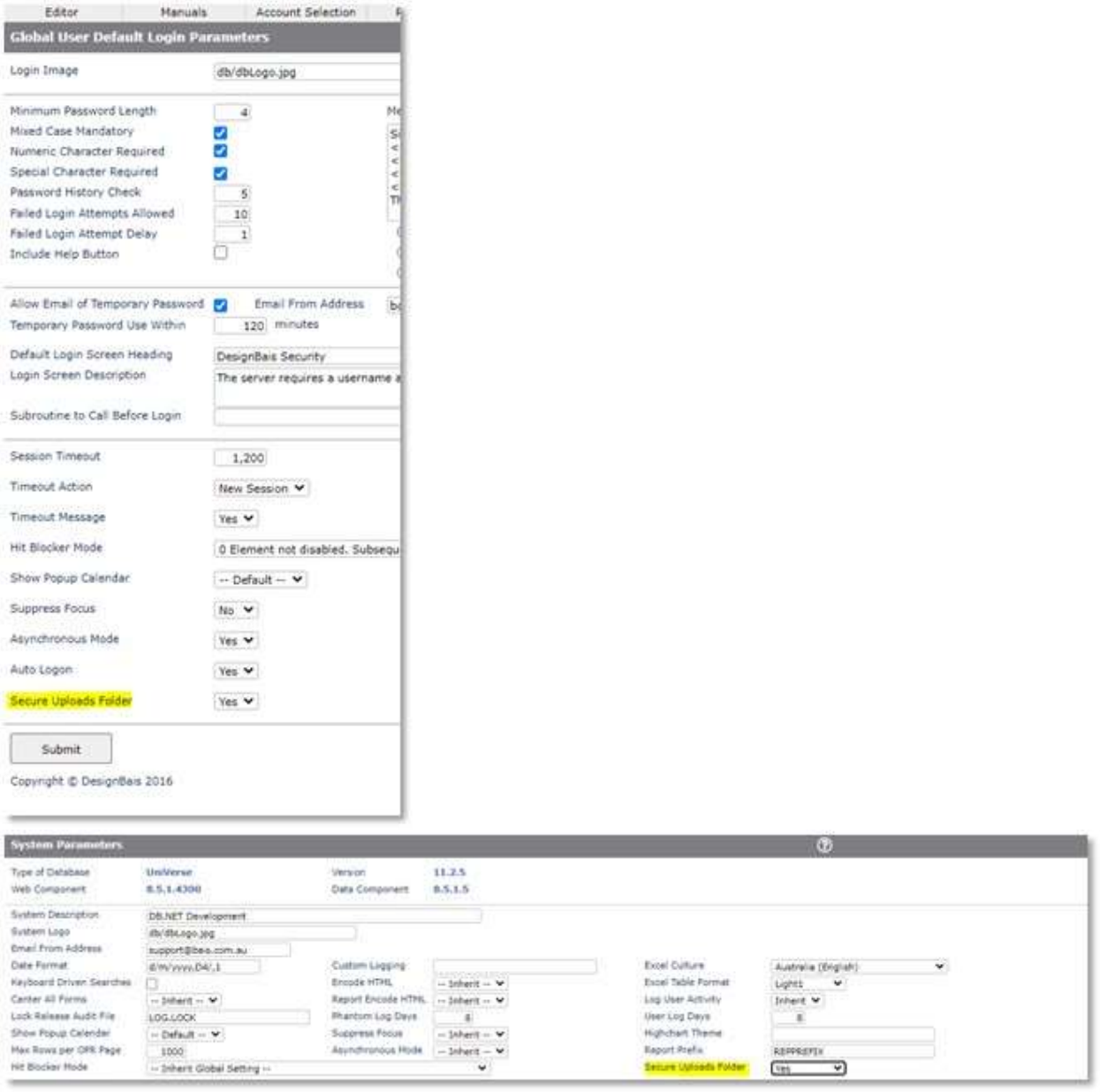
1. Add highlighted parts in web.config

```
<!--UPLOADS FOLDER-->
  <location path="uploads" allowOverride="false">
    <system.webServer>
      <handlers accessPolicy="Read,Execute">
        <remove name="myHandler"/>
        <!--REMOVE THE FOLLOWING IF AUTHENTICATED ACCESS TO UPLOADS FOLDER IS NOT REQUIRED-->
        <!--ALSO CHANGE THE PATH ABOVE FROM "uploads" TO THE TARGET FOLDER IF DIFFERENT-->
        <add name="myHandler" verb="*" path="*" type="DesignBaisNET.checkAuth" />
      </handlers>
    </system.webServer>
  </location>
```

Note: When this feature is not used, change web.config as follows.

```
<!--UPLOADS FOLDER-->
<location path="uploads" allowOverride="false">
  <system.webServer>
    <handlers accessPolicy="Read">
      <remove name="myHandler"/>
      <!-- REMOVE THE FOLLOWING IF AUTHENTICATED ACCESS TO UPLOADS FOLDER IS NOT REQUIRED-->
      <!-- ALSO CHANGE THE PATH ABOVE FROM "uploads" TO THE TARGET FOLDER IF DIFFERENT-->
      <!--<add name="myHandler" verb="*" path="*.*" type="DesignBaisNET.checkAuth" />-->
    </handlers>
  </system.webServer>
</location>
```

2. In DesignBais, set "**Secure Uploads Folder**" option to *Yes* in Global Login Parameters and/or in System Parameters. The *Inherit* option in System Parameters allows individual accounts to use the setting from the global parameters.



3. While this feature is in use, the following error will be displayed when an unauthenticated user tries to access a static file in the *uploads* folder:

4. In Responsive Design the application needs to set a flag to allow access to the *uploads* folder. The flag can be set by either of the following methods:
    - Set SESSION.REC<DBISS.AUTHENTICATED> = “Y”
    - CALL DBI.G.AJXCMD(“AUTH”,”Y”)

5. Authentication for individual users can be controlled by the *Uploads* Access flag in User Maintenance. By default each user will inherit the setting from System Parameters. To allow a user to access the uploads folder set the flag to *Yes*. Note that the user setting will override the logon form setting meaning that access can be switched off or on for a particular user without setting the SESSION.REC attribute.

6. If the *Secure Uploads Folder* flag is turned on then developers must use DBMAIL to email reports to recipients, since, when using DBI.G.SENDEMAIL, reports are sent as links and the recipients will not be able to open the reports.

7. Note that if authentication is on in the web.config file for the website:

```
<add name="myHandler" verb="*" path="*" type="DesignBaisNET.checkAuth" />
```

   then access to the uploads folder will be restricted for **all** users unless either the Global or Systems Parameter *Secure Uploads Folder* parameter is set to *Yes*.

   The *Uploads Access* flag in User Maintenance is only visible when the Global or Systems Parameter *Secure Uploads Folder* parameter is set to *Yes*.

8. To summarise authentication can be provided via:
    - the DesignBais login form
    - a custom login form provided by the application
    - setting the user’s *Uploads Access* flag to *Yes*
    - if a user enters DesignBais using a url with an *ac* code
      (such as *http://192.168.199.194/dbnet/?ac=garb*)
      then the *Uploads Access* flag for this user must be set to *Yes* to allow this user to access the *uploads* folder

Cached Images – Force Refresh

The DesignBais website can force the refresh of images using the following entry in the web.config file:

- To enable revalidation on each request:

```
<httpProtocol>
   <customHeaders>
     <add name="cache-control" value="no-cache" />
   </customHeaders>
</httpProtocol>
```

- To disable caching altogether:

```
<httpProtocol>
   <customHeaders>
       <add name="Cache-Control" value="no-store" max-age=0 />
   </customHeaders>
</httpProtocol>
```

Chapter 7 – Selection Process

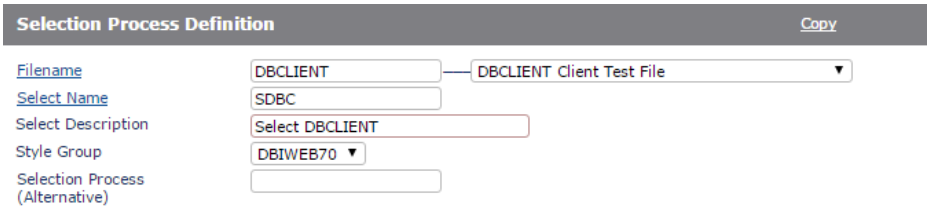
Selection Process

The selection process form provides a very simple way to create complex search processes that deliver a superior user interface and selection performance.

The selection processes reside in the file DBISELECT. The form definition for the selection process resides in the file DBIFORMS. The key to both files is *Filename*SelectName.* To add a selection process to a form the reference required is *Filename_Formname* where *Formname* is the same as the *SelectName*.

There are a number of sections to the Select Process Definition form. In this document they have been separated for simplicity.

Section 1:



Prompts

Filename

Enter a name for a file that is known to DesignBais.
You may also use the dropdown list to select a filename.
Click on the Filename hyperlink to invoke a search for files that are defined to DesignBais.

Select Name

Provides a name for the selection process. You should not include characters that are not valid within keys in your database platform. There can be no underscores in selection names. Otherwise, names of selections are not restricted.

Click on the Select Name hyperlink to invoke a search of selection processes for the selected file.

Select Description

The Select Description will be used as the heading on the created search form.

Style Group

Determines the style (colors and fonts) that are used on the form. Note that the style group will determine the form of the paging buttons for the select. By default the compressed buttons are used:

Checking the *Select Includes Page Buttons* option in Style Group Definition causes the previous paging button style to be used:

Select Process (Alternative)

Is used to define a subroutine that will be used instead of the defined selection statement.

This is useful when you want to perform a selection type that DesignBais does not handle by default. An example may be that you wish to perform a SQL select or similar, which DesignBais does not yet support.

Your BASIC subroutine would be called with the following common variables assigned.

PROCESS.EVENT = "CREATE SELECTION"
PROCESS.EVENTSOURCE = *Filename_Formname*

The common variable DBRETURN.SELECT (*SelectNumber*) is used to return a select list back to the DesignBais selection processor to format the display.

Example of a selection statement that could be used in an Alternate Selection Process:

```
EXECUTE "SSELECT DBCLIENT BY DBC.CLIENT CODE"
        CAPTURING MSG RTNLIST DBRETURN.SELECT(1)
```

At this time the *SelectNumber* should always be '1'. This coincides with the Select Number field on the form.

Each MV environment passes and returns lists in a different format. In some cases the lists are attribute-delimited dynamic arrays and for other systems lists are internal structures. Be sure the assignment to DBRETURN.SELECT is a valid list variable for deployment platforms, not just for development.

In UniData the RTNLIST command must precede the CAPTURING / RETURNING commands:
EXECUTE CMD RTNLIST DBRETURN.SELECT(1) CAPTURING DETS RETURNING ERRS

In UniVerse the following structure will compile:
EXECUTE CMD CAPTURING DETS RETURNING ERRS RTNLIST DBRETURN.SELECT(1)

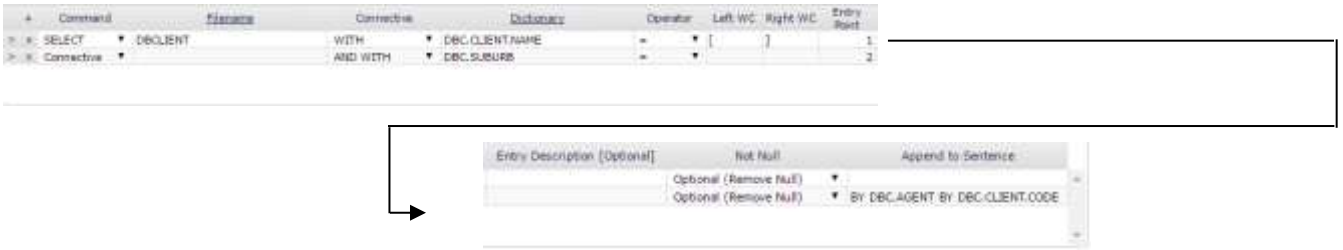
Note that, when using a basic subroutine to select from a file different to the file on which the selection process is defined, it may be necessary to use:

DBSWITCHSELECTFILE = "select.process.filename"
DBSWITCHSELECTFILETO = "select.data.from.filename"
DBSELECTAPPEND = " USING DICT select.process.filename "

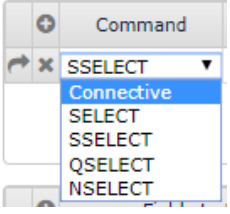
Buttons

Copy Selection — Used to copy existing selection processes.

Section 2:



Command



This prompt is used to determine which type of select command you wish to use. The **Connective** option is used to join multiple conditions within a single selection statement.

Eg SSELECT DBCLIENT WITH DBC.CLIENT.NAME = "[GLASS]"
{Connective} AND WITH DBC.SUBURB = "New York"

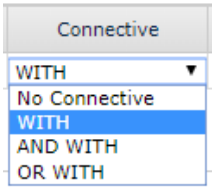
The results from each statement are additive. The final list of keys will be used to render data from the file associated with this selection process.

Filename

Is the file that is to be used for the select. You can click on the Filename hyperlink in the header to view the available files. See also the section on Switching Files at Runtime and the variables DBSWITCHSELECTFILE and DBSWITCHSELECTFILETO.

Connective

The **Connective** prompt allows you to create a conditional link between filename and the dictionary field.



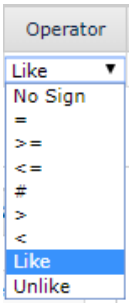
Eg. SSELECT DBCLIENT WITH DBC.CLIENT.NAME *{rest of sentence}*

Dictionary

Is used to define the dictionary definition that is used for conditional selections. You may click the Dictionary hyperlink to list the available dictionary fields (defined DesignBais Field Properties) for the file being selected.

Operator

The Operator prompt assigns the type of matching to be performed in the sentence.



Eg. SELECT DBCLIENT WITH DBC.CLIENT.NAME = *{rest of sentence}*

When using the *Like* or *Unlike* options the *Left* and *Right Wild Card* fields will normally be left blank. The value that is used as the selection criteria can contain the "…" string in order to allow the search string to match any part of the target field content.

For example "F…T…" will match with field values like "Further]" or "Fiscal Tender" (provided the search field is case insensitive).

Left Wild Card

The Left Wildcard (if used) provides a prefix to user entry. Generally the prefix is the database wildcard character (usually '[').

Eg. SSELECT DBCLIENT WITH DBC.CLIENT.NAME = "*[{User Entry}*"

Right Wild Card

The Right Wildcard (if used) provides a suffix to user entry. Generally the suffix is the database wildcard character (usually ']').

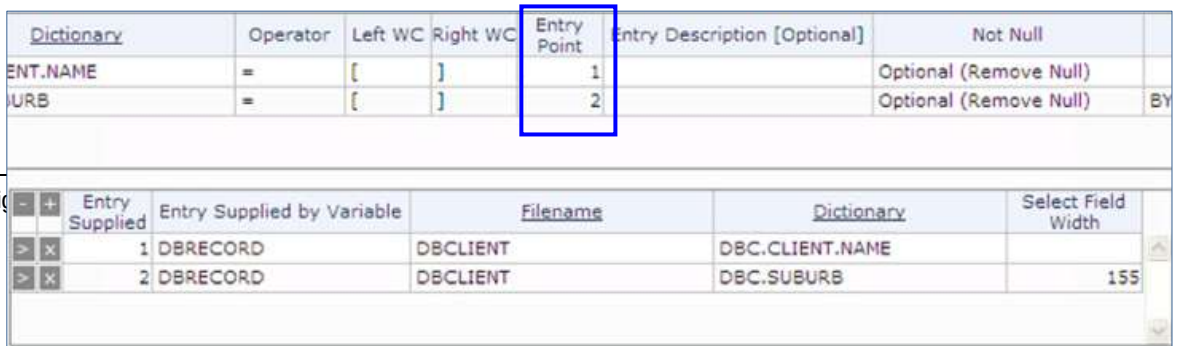
Eg. SSELECT DBCLIENT WITH DBC.CLIENT.NAME = "*{User Entry}]*"

Entry Point

The Entry Point defines a number representation for each input field on the selection form. See below for a description of how DesignBais processes selection criteria that do not specify an "Entry [point] Supplied".

In the example below, Entry Point 1 and 2 (blue surrounds) represent two input fields on the search form. The first being *Client Name* and the second *Suburb.*

The validations and length of fields are supplied by the Entry Supplied section (green surrounds). This helps to ensure that input fields on a search form have the same characteristics as fields placed on a normal form. If the Entry Supplied section is not populated, and Entry Point numbers are entered, then the selection fields that appear on the search form will use the validation and length properties of the corresponding field defined by the Filename and Dictionary entries on the same row and to the left of the Entry Point.



Dictionary	Operator	Left WC	Right WC	Entry Point	Entry Description [Optional]	Not Null	
ENT.NAME	=	[	]	1		Optional (Remove Null)	
URB	=	[	]	2		Optional (Remove Null)	BY

Entry Supplied	Entry Supplied by Variable	Filename	Dictionary	Select Field Width
1	DBRECORD	DBCLIENT	DBC.CLIENT.NAME	
2	DBRECORD	DBCLIENT	DBC.SUBURB	155

Copyright © 2021 Desig

The resultant Search form:

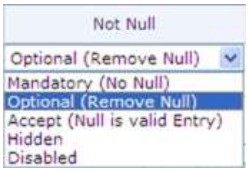
At run-time, values for Client Name and Suburb are passed from the data entry form to the selection form, pre-loading the values for the selection process. These values can be used or discarded by the user for selections.

Entry Description

If an entry point is not supplied by a field property, a description can be entered to support the input. If the entry point is added, the description entered will replace the description in the field property

Not Null

This field defines how the line in the selection sentence is handled after user input.



Mandatory — The user must enter a value in the input field corresponding to the line in the sentence

Optional — The user has the choice of entry in the field.

In the examples above the selection sentence would normally read:

```
SSELECT DBCLIENT WITH DBC.CLIENT.NAME = "[User Entry]"
      AND WITH DBC.SUBURB = "[User Entry]"
```

If the user does not enter a value in the suburb prompt (which is optional), the 'AND WITH DBC.SUBURB = "[*User Entry*]"' would be dropped from the sentence by the DesignBais selection processor. The end result would be:

```
SSELECT DBCLIENT WITH DBC.CLIENT.NAME = "[User Entry]"
```

This provides a powerful way to customise the selection statement "on the fly" based on what the user inputs.

Accept — Null is a valid entry by the user. You must set the 'Not Null' value to this if you want to perform a selection statement like:

```
SELECT MYFILE WITH MYDICTIONARY = ""
```

Hidden — The dictionary that has been included in the selection statement is not to appear as an input field on the Search form.

Disabled — The dictionary that has been included in the selection statement is to appear on the Search form, but the end-user is not able to modify its contents.

Append to Sentence

Add extra criteria to your sentence for sorting or unique processing etc. In the above examples there are two lines of dictionary assignments. For the Append to Sentence function to perform correctly, the sort criteria (as above) must be entered onto the second line.

The resultant sentence would be:

```
SSELECT DBCLIENT WITH DBC.CLIENT.NAME = "[User Entry]"
       AND WITH DBC.SUBURB = "[User Entry]" BY DBC.CLIENT
```

To hardcode a literal into the selection criteria, append the literal with no Entry Point information. For example, to only show the clients in the state of Ohio, use Dictionary DBC.STATE, Operator =, null for the following fields, and “OHIO” in quotes for the Append to Sentence. This results in:

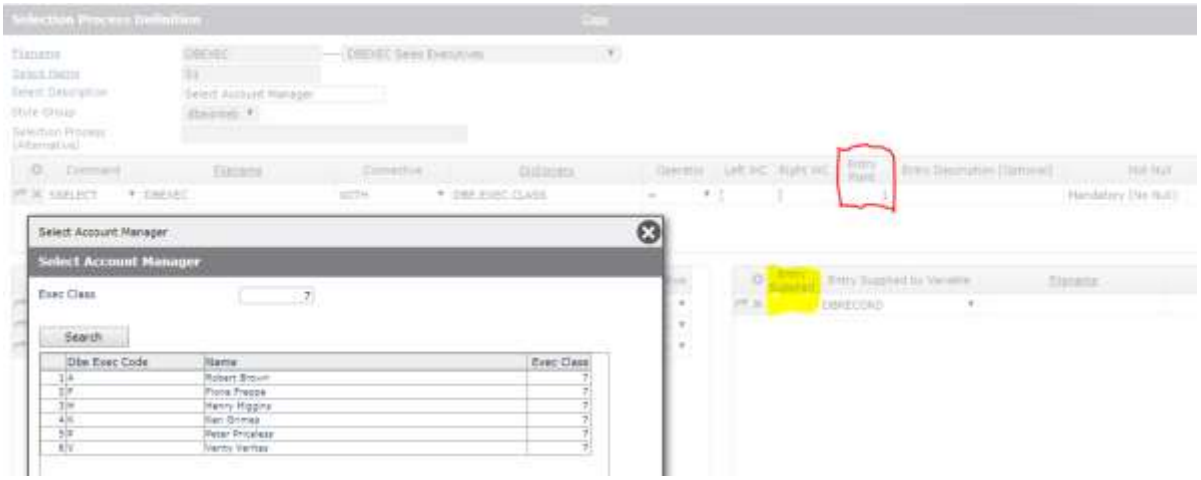
```
SSELECT DBCLIENT WITH DBC.STATE = "OHIO"
```

No Entry Point

DesignBais will check field names in the select statement for entry fields without an ‘Entry [Point] Supplied’ to pick up Field Type (Alpha, Date, Numeric) and Number of Decimals. This data is also stored in the report “RUN.” Forms.

Note that you must have a value in ‘Entry Point’ in order to trigger the display of a selection field in the selection form.

The selection field ‘Exec Class’ then displays in the selection form, provided there is no value in the ‘Entry [Point] Supplied’ field.



Section 3:

	Fields to Display	Return Key	Provide Refinement	Display Width Pixels	Encode HTML	Case Sensitive
	DBE.EXEC.CODE	As Displayed	No	150	-- Inherit --	-- Inherit --
	DBE.EXEC.NAME	As Item Key	Yes	350	No	Yes
	DBE.EXEC.CLASS	No	Yes	70	Yes	No

Prompts

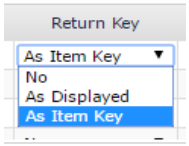
Fields to Display

Used to control what fields are displayed in the search results section of the search form. For platforms that support I-Type descriptors, if the Create Correlative as an IType check box in Field Properties is checked, the compiled I-Type will be used to display the field in the selection process.

Click the Fields to Display hyperlink to view a search form containing the definitions for the selected file.

Return Key

Determines the key that gets passed back to the calling process.



No — The displayed field is not passed back to the calling process

As Displayed — The field gets passed back to the calling process as displayed on the search form.

As Item Key — The displayed field is not passed back to the calling process. Rather, the key from the selection is returned. This is useful to pass back a key without the user seeing the actual key value.

Provide Refinement

Controls whether the field will appear in the **Refine Search** section of the search form. When users click the Search button in selection forms, a new selection is initiated using the command defined here. Fields flagged to Provide Refinement are added to the bottom of search forms. By clicking the provided Refine button, the refined values are applied to the currently active selection. Selections can be refined as many times as desired.

Display Width Pixels

Controls the width (in pixels) of the display field in the search form. If the sum of all display widths is greater than the form width, data scrolls horizontally within the form width.

Encode HTML

Display fields containing HTML elements should be encoded to prevent XSS injection attacks. Encoding HTML means converting characters that have meaning in HTML to their display only string.
For example the 'less than' character < is encoded as & l t ; (without the spaces). This means that any HTML will display as entered but will not be active in the browser.

Encoding may be set on Display Fields or for the entire Select. If a Display Field is set to Inherit (the default action) then the Select Encode HTML value is used (see Select Encode HTML field described below). If the Select Encode HTML value is Inherit than the System Parameter setting will be applied; if the System Parameter value is Inherit then the Global Setting will be applied.

It is recommended that HTML Encoding be turned on for all fields except for fields where active HTML elements are used.

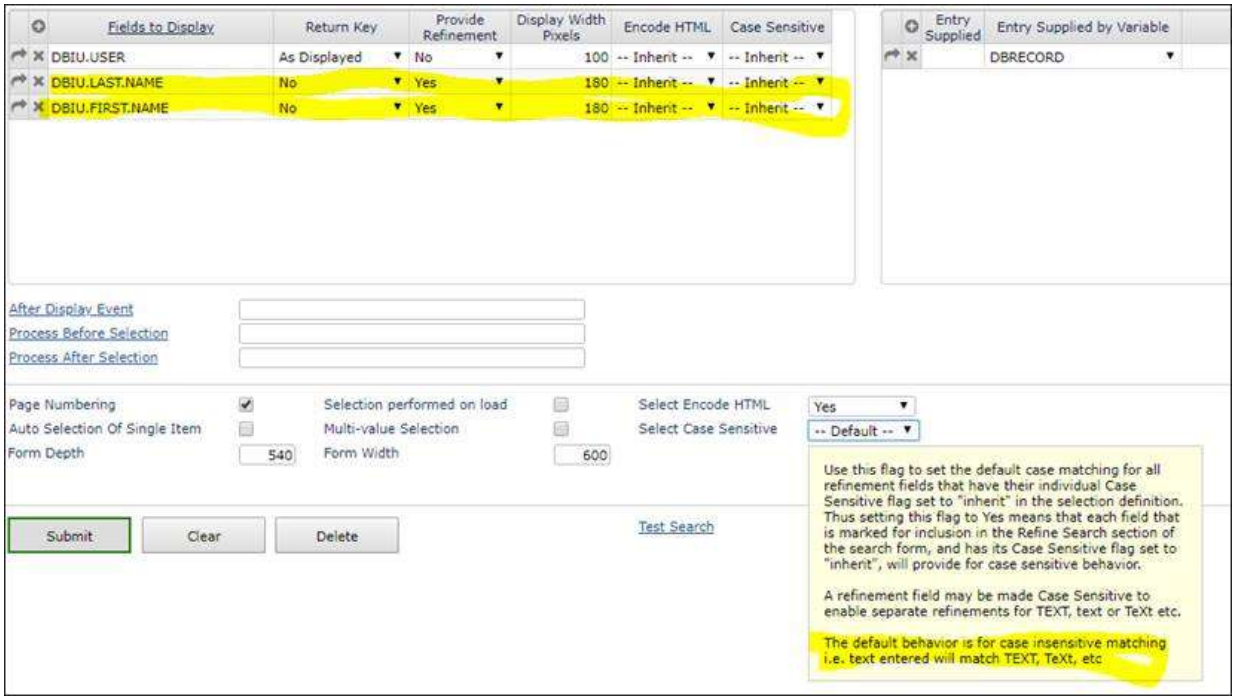
Case Sensitive

Use 'Inherit' to use the case sensitive setting set for the selection process as a whole (see Select Case Sensitive field described below). The default behavior is for case insensitive matching which means that the case of the text entered will not effect the selection result. Entering 'Text' or 'TEXT' or 'TeXt' will serve to find a string containing 'TEXT' or 'text' or any combination of case.

Setting this flag to No results in this behaviour. To enable separate refinements for 'TEXT', 'text' or 'TeXt' set this flag to 'Yes'.

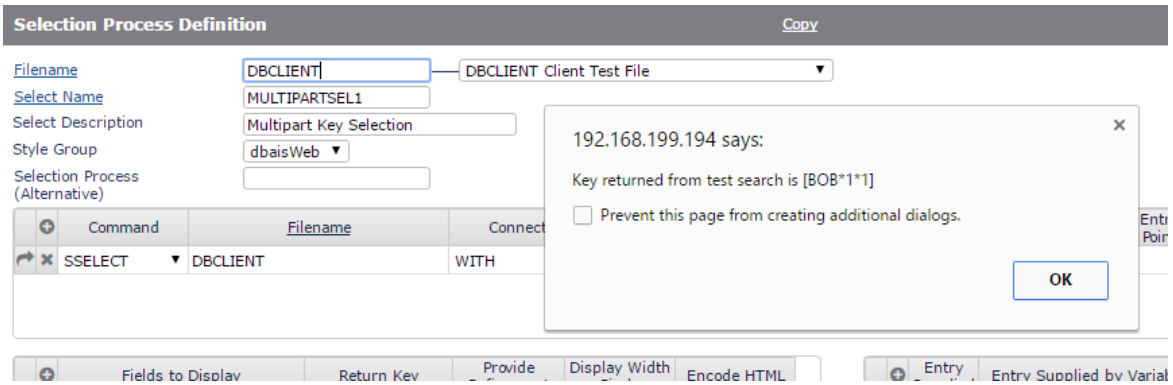
The Case Sensitive flags on the Select Process itself relate to the "*Refine*" fields at the base of the **search form**, NOT to the selection fields in your select.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

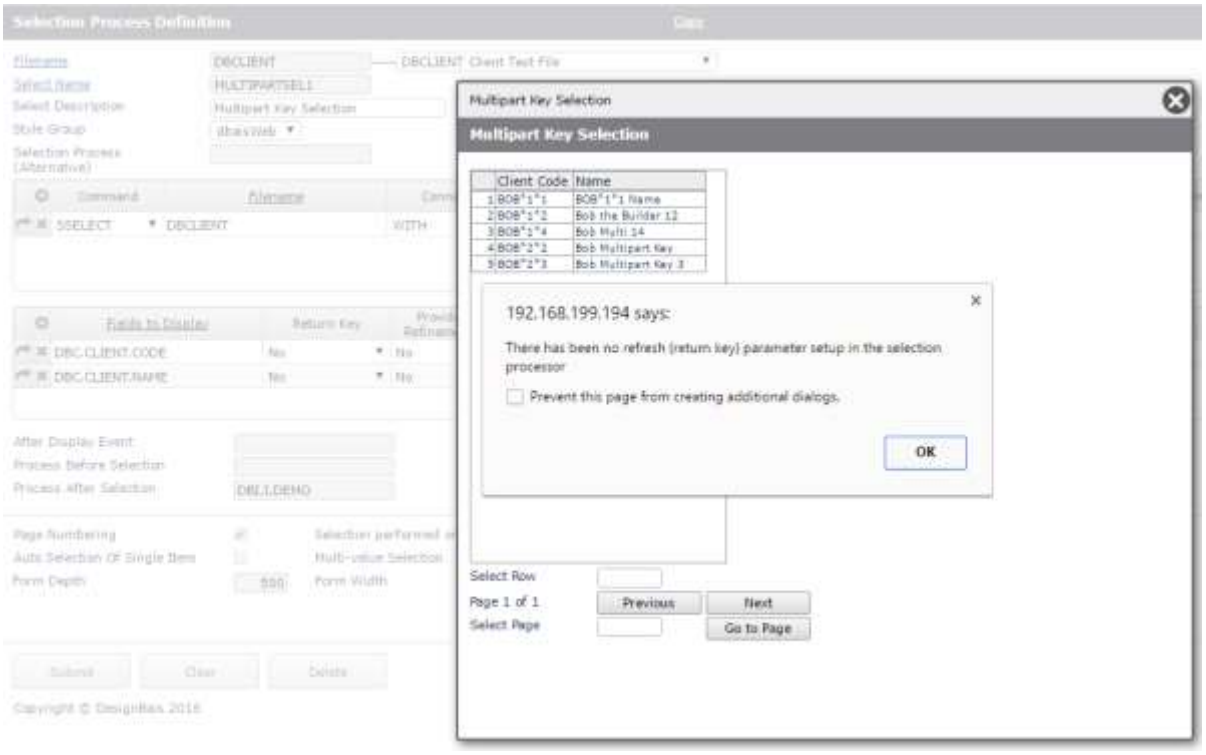


The “Refine” fields are marked in yellow below:

When running a selection in test mode using the Test Search button the selected key will be displayed as shown:



If the “Return Key” column of the Fields to Display grid has not been entered (that is all rows show “No”) then a warning message is displayed as shown below:



The number of fields in the Refine Search section of the selection form impacts the number of selection result rows displayed.

The height of the On-form Report displaying the selection results is calculated from the form height less the height of all the other form elements. Results are displayed with both paging and a scrollbar if required.

The number of rows per page is determined as:

Rows per page = INT(OFR display depth / 18) – 1

Section 4:

Entry Supplied	Entry Supplied by Variable	Filename	Dictionary	Select Field Width
1	DBRECORD	DBCLIENT	DBC.CLIENT.NAME	
2	DBRECORD	DBCLIENT	DBC.SUBURB	155
	DBRECORD			

Entry Supplied — Assigns a numeric value for each entry point. Numbers in this section should match with those in section 2. In certain cases you may require input fields on the search form that do not participate in the selection sentence. You may add extra Entry Supplied values for this purpose. This is particularly useful when selection processes are being used in reports.

Entry Supplied by Variable

In cases where the developer wishes data from the form to be automatically loaded into a selection form, nominate the variable that contains the required value.

Filename — Enter the filename that is referenced against the required field property.

Click the Filename hyperlink to display the search form for files.

Dictionary — Enter the name of the dictionary (field property) that is to be used to supply the details for the input field on the search form.

Click the Dictionary hyperlink to display the search form for dictionaries (field properties) for the defined file.

Select Field Width — By default, the width of input prompts on a search form is derived from the Field Property. If this field has a value it with override this default behaviour.

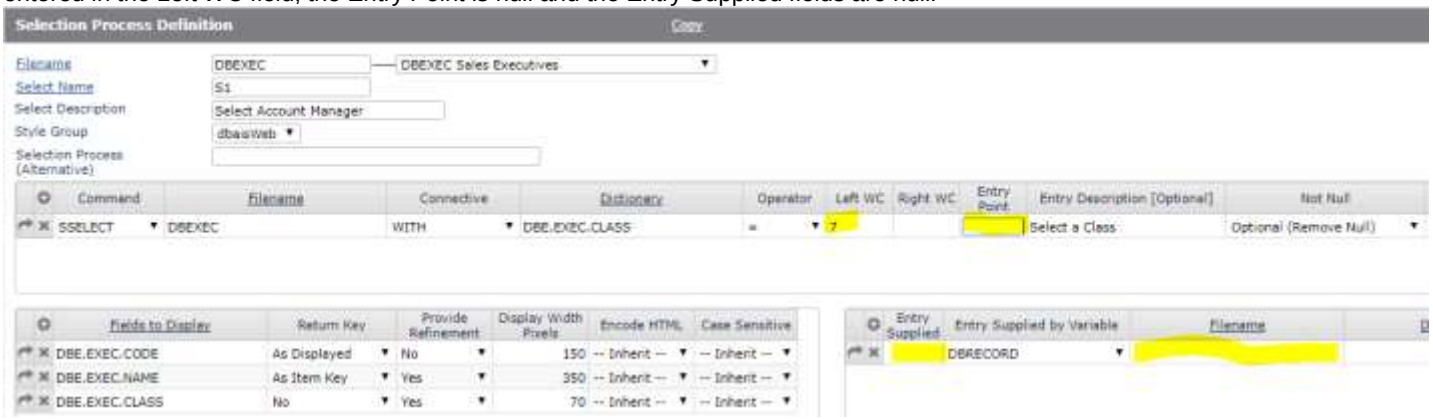
Selecting fields with specific values

The example below shows how to select a field which is null.

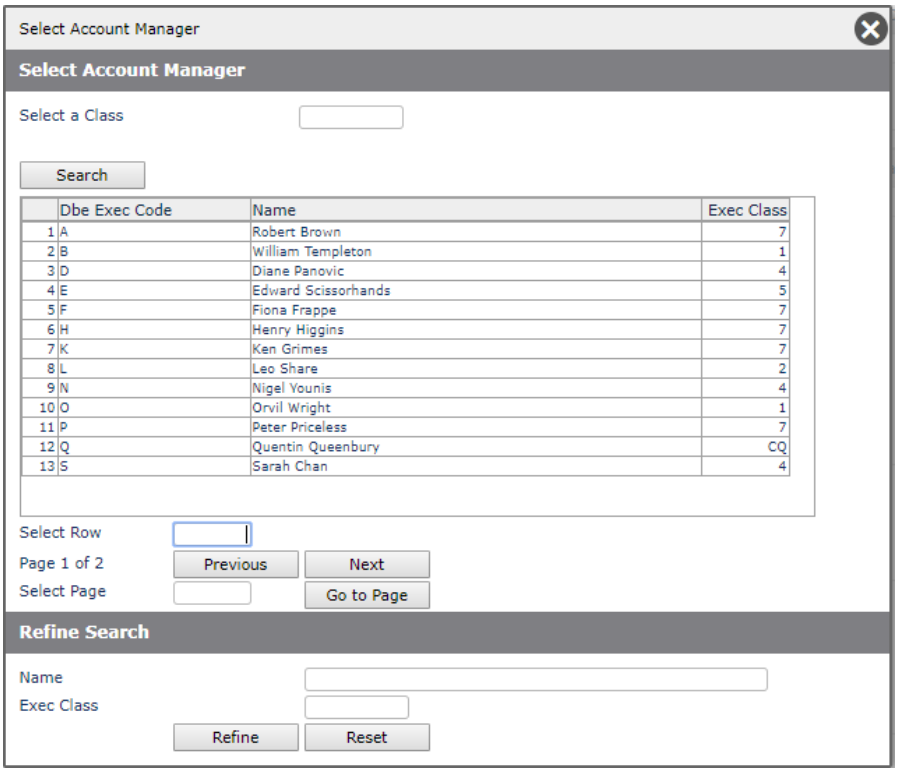
Command	Filename	Connective	Dictionary	Operator	Left WC	Right WC	Entry Point	Entry Description [Optional]	Not Null
SSELECT	DBCLIENT	WITH	DBC.CLIENT.NAME	=					Accept (Null is valid Entry)

This will result in the statement 'SSELECT DBCLIENT WITH DBC.CLIENT.NAME = "" '.

The following example shows how to select Sales Executives with a Class Code of 7 (for example). Note that the required value is entered in the Left WC field, the Entry Point is null and the Entry Supplied fields are null.

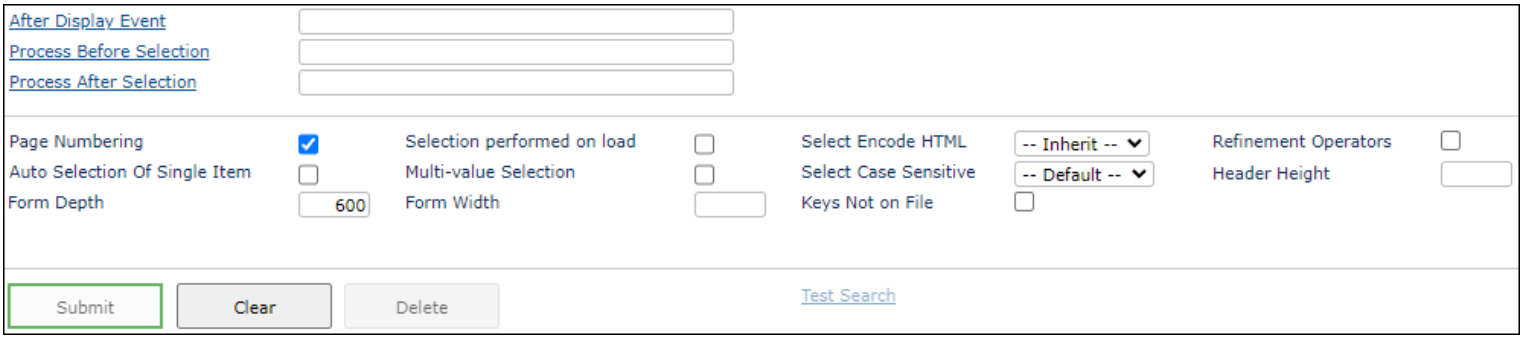


Adding an Entry Point to the selection (but leaving the Entry Supplied fields blank) results in a selection as shown below:



Note that the select field appears in the Selection form but the particular value, in this case 7, is not utilised.

Section 5:



After Display Event — This slot allows the developer to call an After Display Event for the search form. This is useful to provide default values to the search form. The PROCESS.EVENT is "AFTER DISPLAY". The SCREENROOT will be the search form.

Process Before Selection — Allows for a subroutine call to further validate user input.

PROCESS.EVENT = "BEFORE SELECTION"
PROCESS.EVENTSOURCE = Select Form Name (*Formname* only, not *Filename_Formname*)

You can set IERR.TEXT to a non-null value to interrupt the search and display an error message.

Process After Selection

Allows for a subroutine to be executed when the user selects an item from the selection form. The value to be returned to the calling form (as specified in the Return Key field) is available in DBVALUE.

The parameters for the called subroutine are:

PROCESS.EVENT = "AFTER SELECT"
PROCESS.EVENTSOURCE = Selection Form Name

Page Numbering

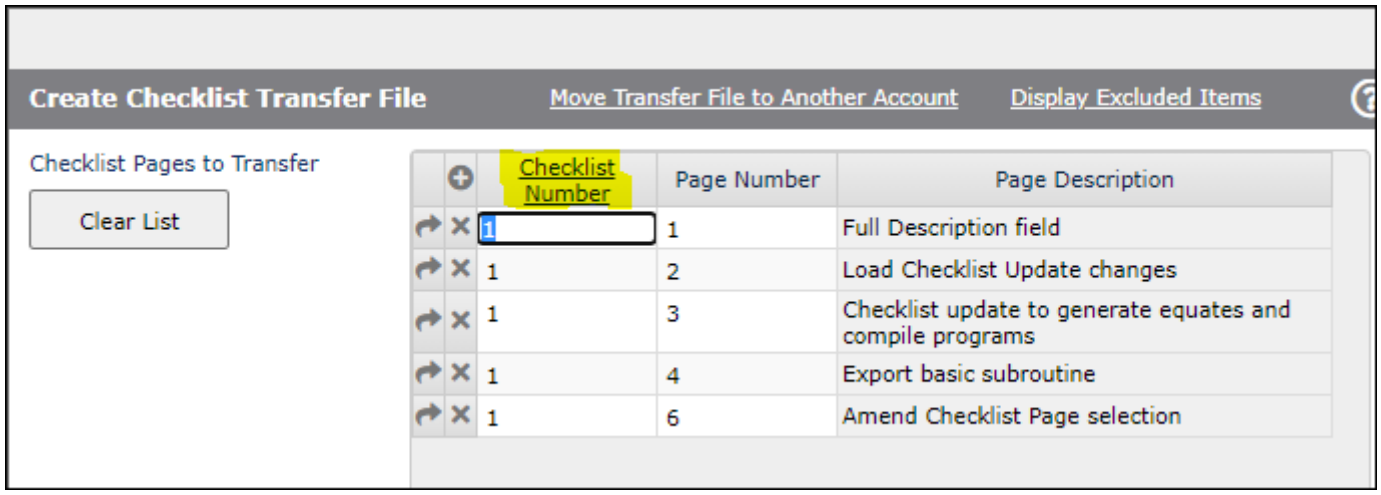
When checked will place Paging Control buttons on the search form.

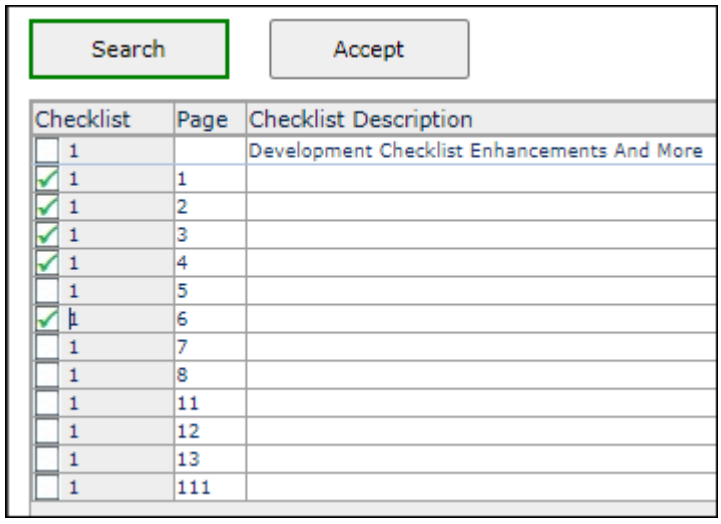
Auto Selection Of Single Item

If this flag is set, a selection returning a single record will automatically return the item to the calling process. If this flag is not set, the selection report will be displayed.

Multi-value Selection

The field indicates whether the response from a selection will populate a multi-value field. If true (checked), the selection form will allow for multiple selections by the user and an 'Accept' button will be placed on the search form. Selection can be made by clicking the check box in the first display column or by entering the row number in the Select Row field below the display. The selection method is controlled by the setting on the Style Group.





Selection performed on load

When checked will run the selection as part of the search form load process. This is very useful when there is no criteria that requires input from the end user to complete the search.

When the search form is loaded the selection process will be invoked and the resultant selected items displayed.

Form Depth — Controls the depth (height) of the search form (in pixels).

Form Width — Controls the width of the form (in pixels)

Select Encode HTML — This can be Inherit, Yes or No. Inherit will inherit the Encode HTML from the System Parameters setting. Display fields containing HTML elements should be encoded to prevent XSS injection attacks.

Encoding HTML means converting characters that have meaning in HTML to their display only string.

For example the 'less than' character < is encoded as & l t ; (without the spaces).This means that any HTML will display as entered but will not be active in the browser.

Encoding may be set on Display Fields or for the entire Select. If a Display Field is set to Inherit (the default action) then the Select Encode HTML value is used. If the Select Encode HTML value is Inherit than the System Parameter setting will be applied; if the System Parameter value is Inherit then the Global Setting will be applied.

It is recommended that HTML Encoding be turned on for all fields except for fields where active HTML elements are used.

Select Case Sensitive — The default behavior is for case insensitive matching which means that the case of the text entered will not effect the selection result. Entering 'Text' or 'TEXT' or 'TeXt' will serve to find a string containing 'TEXT' or 'text' or any combination of case. The "Default" setting is provided for backwards compatibility and means that this flag is set to null and leaves the system to behave as it always did.

To enable separate refinements for 'TEXT', 'text' or 'TeXt' set this flag to 'Yes'.

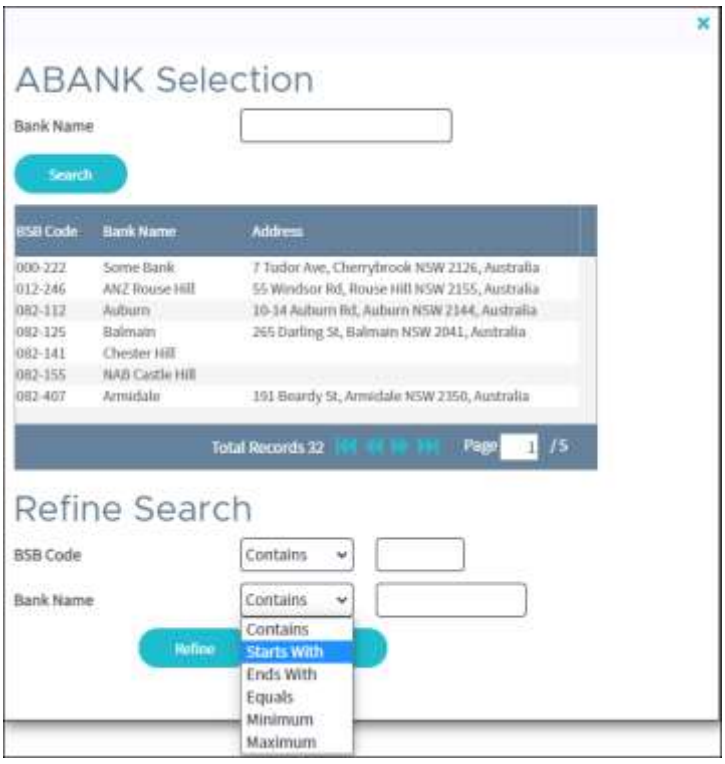
Keys Not on File — Use this flag to display keys that are not on a DesignBais file e.g. after a QSELECT. Developers can define DesignBais derived fields to be used in conjunction with these keys in order to display, for example, a description for each key, in the Fields to Display list. In this case DesignBais does not use the derived field parent field to force the calculation of the field value. Rather the selection process passes in the qselected keys. Each key is used to drive the calculation of the derived field. If the derived field defines a *Parameter* in conjunction with the *Subroutine to Derive* then it will be passed in with the selected key appended and separated by a pipe character, as shown here:

```
PROCESS.EVENT = "DERIVED"
PROCESS.EVENTSOURCE = Field Name
PROCESS.PARAMETER = Field Parameter | Select Key
```

Refinement Operators — Use this flag to provide a dropdown list of refinement options for all fields in the *Fields to Display* grid that are flagged for *Provide Refinement – Yes*.

The options are:

```
Contains      - traditional option that applies without operator choice "[...]"
Starts With   - string beginning "...]"
Ends With     - string ending "[..."
Equal         - exact match
Minimum       - greater than or equal to  (for numeric fields)
Maximum       - less than or equal to     (for numeric fields)
```



Header Height	The height in px for the header row of the select. Useful to increase the default header height set from the Report Header style on the Style Group if more than the default of 32px. This will adjust the number of rows displayed per page or simply add some space to the heading text.

Buttons

Submit	Will save the selection process in the file DBISELECT. The key to the file is *Filename*Select Form Name.* The resultant search form is written to the DBIFORMS file with the same key structure.
Clear	Will clear the form and not preserve any changes.
Delete	Will delete the selection process from the DBISELECT file and also delete the search form from DBIFORMS.
Test Search	This button will run the search process in test mode. Selections are run as local forms from DBISESSIONS. In test mode the SESSION.ID, rather than the form name, is used so as not to overwrite the current runtime version. In development if you display the screenroot you will see the session id rather than the selection process name.

Selection Field Focus

When a Selection Process runs field focus will be set in the following order:

- the first *input* field for search criteria
- the Select Row for keyboard driven searches
- the first Refinement input field
- the paging control go to page

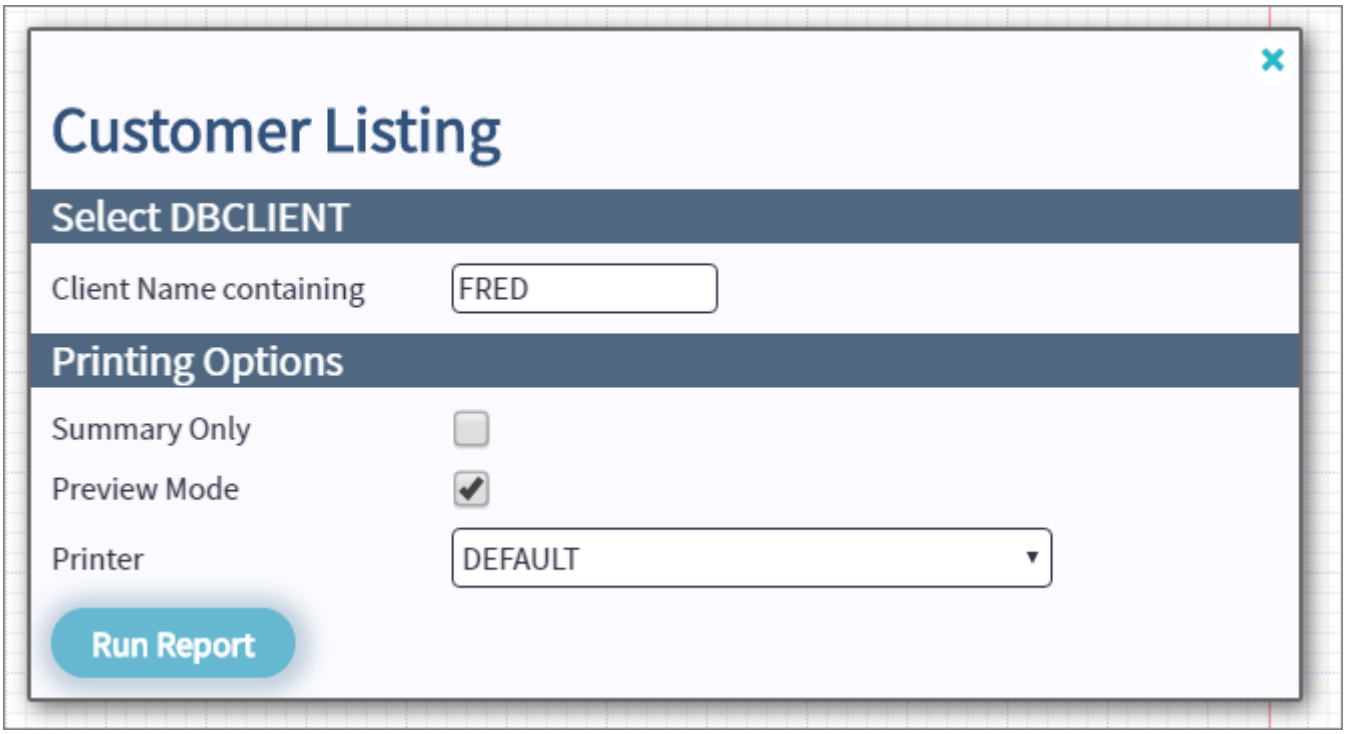
This mean, for example, that for selections with no search criteria fields and no refinement fields, focus will be on the go to the paging control field.

Selection Process Style Group

The selection form that appears when a selection process is invoked is built by the DesignBais engine. This occurs when the *Submit* button or the *Test* Search button is pressed in the Selection Process Definition form.

As of release 8.3.1.3 this process now uses the properties of the style group specified in the *Style Group* field of the selection process to format the selection form. Previously DesignBais applied fixed gaps between form elements and fixed sizes to buttons and other form elements.

The example below shows how a selection form can look when built with a particular style group.



Chapter 8 – Forms Designer

Forms Designer

The Forms Designer is a graphical tool for the creation of browser-based forms.

W3C (Ajax) Compliant forms

W3C compliance is a standard for common compliance across different browser types.

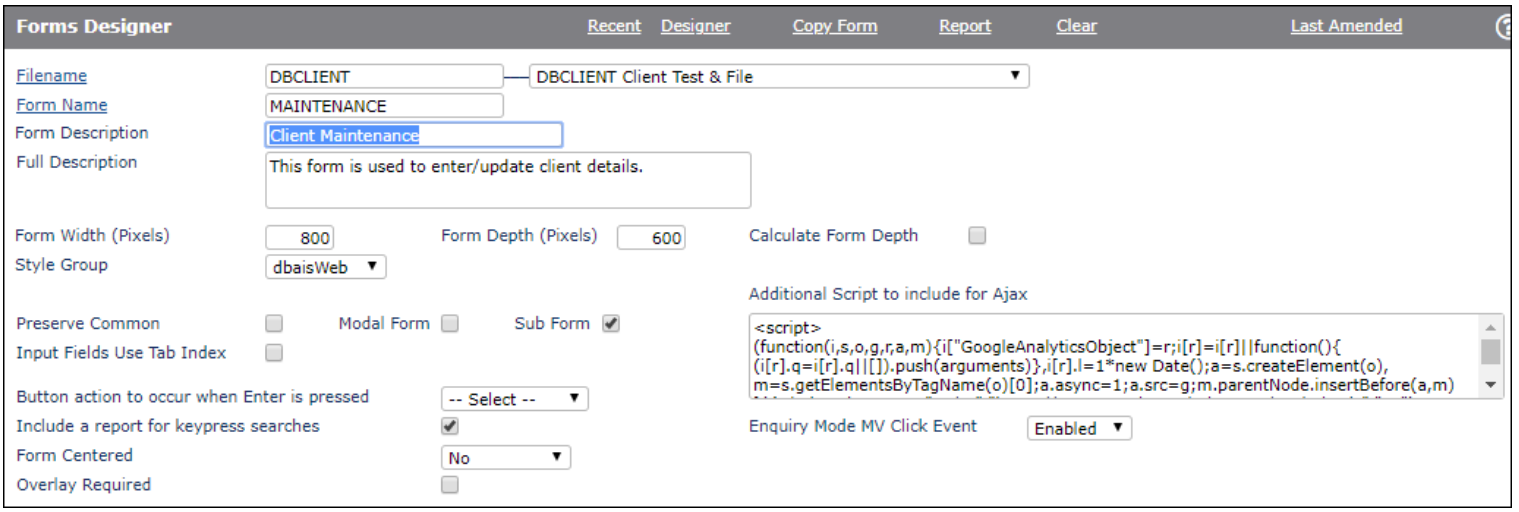
The DesignBais interface provides for before and after-field processing, cursor positioning from server events, full relationship-based read and write processing, multi-form processing, layered form processing, extremely high performance interface, and server-based logic. In other words all of the things that developers and end-users expect from an application development environment, with the bonus of zero client-side deployment.

Multi-value developers will find full support for multi-value grids, including off-form associations and multi-value event processing.

Developers can use the forms designer to generate forms in W3C Ajax mode, this will provide post-field validation, and other browser events without the requirement of a form-submit, which is used in the standard W3C compliance set.

There are a number of sections to the forms designer main form. In this document they have been separated for simplicity.

Section 1a:



Prompts

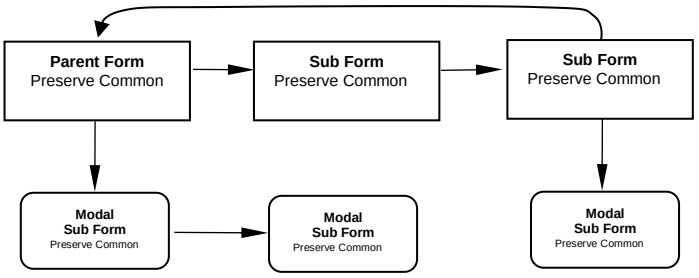
Filename	Enter a name for a file that is known to DesignBais. You may also use the dropdown list to select a filename. **Click on the Filename hyperlink to invoke a search for files that are defined to DesignBais.**
Form Name	The name of the form to create or modify. You should not include characters that are not valid within keys in your database platform. Otherwise, names of forms are not restricted. Characters specifically prohibited by DesignBais are: _ (underscore), – (hyphen), * (asterisk), ~ (tilde) and | (pipe). **Click on the Form Name hyperlink to invoke a search of forms for the selected file.**
Form Description	Enter a short description to support the form. This description will display on search forms
Full Description	The full description of the form will be available on menu-help and DesignBais help files.
Form Width (Pixels)	The width of the form in pixels. For commercial-grade applications it is recommended to design for 1024 x 768 pixels screen size as a minimum.
Form Depth (Pixels)	The Form Depth (height) in pixels. See Header Form Overlay section of this manual.
Calculate Form Depth	If checked, and the Form Depth is not set to zero, then the form depth will be calculated when the form is saved with the Save button in the Designer grid. The Form Depth will only be amended if the actual depth has increased not decreased. In order to amend the Form Depth either up or down simply uncheck then recheck this flag.
Style Group	The Style Group is used to set the standard look and feel properties of the form. All DesignBais forms use the style group DBIWEB. As a standard, DesignBais is shipped with DBIWEB, DBIREP (Style group for reports) and DBI.
Preserve Common	When the form is part of a multi-form set this field must be ticked. This ensures that when the form is invoked as part of a form-set that common variables are not re-set.
Modal Form	This field indicates that the form will be loaded as modal. This means that when loaded the form will over-lay the base (main) form as a window. The based form cannot be accessed until the modal form is closed.

The size of the modal form (modal window) will be the size indicated by the Form Width and Form Depth fields. The position of a modal form can be found in the common variable DBMODALPOS.

Sub Form

This field is used to indicate whether the form is part of a form set. When this field is checked DesignBais treats the form as a child to the calling form. This should always be used in conjunction with the Preserve Common field.

Structure of Sub-forms in a parent/child combination:



Refer to the section **Calling Forms** for further detail.

Additional Script to Include for Ajax

This field is used to define additional script to be added to the standard DesignBais script loaded for a form.

Additional Script to include for Ajax

```
<script src='https://ssl.google-analytics.com/ga.js'
type='text/javascript'></script>
<script type="text/javascript">
```

In the example above script is being added to allow tracking of page activity within google analytics.

Input Fields Use Tab Index

If this item is checked then the sequence used for the input fields on the form will be determined by the tab index. When adding a new field to a form the assigned tab index will be inserted in the correct tab sequence. If this is not checked then the input fields on the form will be sequenced by row and then by column within row.

Deselecting Tab Index use will clear existing tab index settings. Entering Tab Index values in Designer mode will turn this flag on if it is not already ticked.

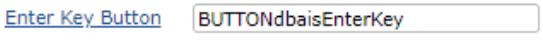
If DBTABINDEX is used to change tab index sequence at run time then the tab index values on the form need to remain unchanged. In this case you can initially set this field on and set a tab index increment of say 10 and save the form. Then by setting the tab index increment back to null or 0 and saving the form the tab index values will retain the increment that was used but will not be re-sequenced when the form is subsequently maintained. DBTABINDEX can then be used to set appropriate tab index values that blend with those on the form.

Tab Index Increment

If Tab Index Increment is set then the value will be used to automatically sequence and increment the tab index for fields on the form. The increment will leave gaps in the tab sequence to allow fields to be re-sequenced in the Tab Index form. If the increment is not set or set to 0 then the tab sequence is not reset.

Button action to occur when enter is pressed

On some forms you may wish to have a button actioned when the user hits the enter key. Select the required button name from the dropdown list. This field is validated against the form, so it is necessary to add the button to the form in the first instance. DesignBais will apply the style defined in the form style group *Enter Key Button* to the button in both designer mode and at run time. This indicates to the user at run time which button will be activated by pressing the enter key.



Include a report for keypress searches

You must have the this field checked if you are implementing keypress searches using the routine DBI.G.GET.DBFINDEX. This routine uses the 11th On-form Report subscript, which is reserved for DesignBais usage, and this flag triggers its setup for use by a form.

Form Centered

Controls if and how the form is centred in the browser window.
Set this field to "Yes" if you want the form to be centred in the browser window.
Setting to "No" will Left Align.
Setting to Inherit will use the System Parameters Center All Forms setting.
If the System Parameters setting is “Inherit" then the General Global Parameters Setting will be applied.

Forms can be moved using the following call in a basic subroutine. After the form has been rendered and in the case of a modal form, centered, the form can be moved:

```
DBAJAXCMD<-1>='document.getElementById("dbBackGround':DBWLEVEL:'").style.top="0px";    '
```

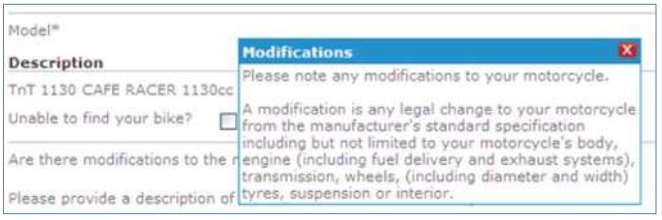
For another example a form can be moved to the top and left based on a hidden-button click event:

```
CASE SCREEN.NO = "D12" AND EVENTSOURCE = "B.MOVE"
   * Position next to DBISTYLEGROUP*D12 grid
   LEFT.POS=600 + DBSIDEMENUWIDTH
   LEFT.SG='document.getElementById("dbBackGround':DBWLEVEL:'").style.left="':LEFT.POS:'px";    '
   DBAJAXCMD<-1>='document.getElementById("dbBackGround':DBWLEVEL:'").style.top="0px";    '
   DBAJAXCMD<-1>=LEFT.SG
```

(Note that these command strings must end with a semi-colon followed by 3 spaces).

Overlay Required

There may be times where help dialog boxes are required on a browser form. If this is required then an overlay field can be added to the DesignBais form at runtime. This is then supported by the DBI.G.OVERLAY subroutine which is used to build the overlay. See documentation for DBI.G.OVERLAY.
Example of usage:

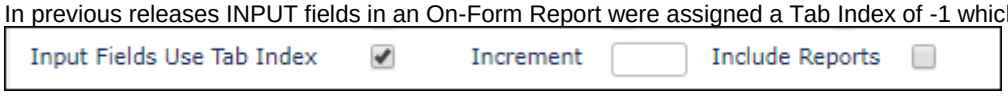


Enquiry Mode MV Click Event

Traditionally DesignBais has left multivalued field "Click event on process after" active in Enquiry Mode. If set, this flag will disable MV clickable columns for this form in Enquiry Mode. Fields can be activated by including them in the list of "Enquiry Only (Input) Fields".

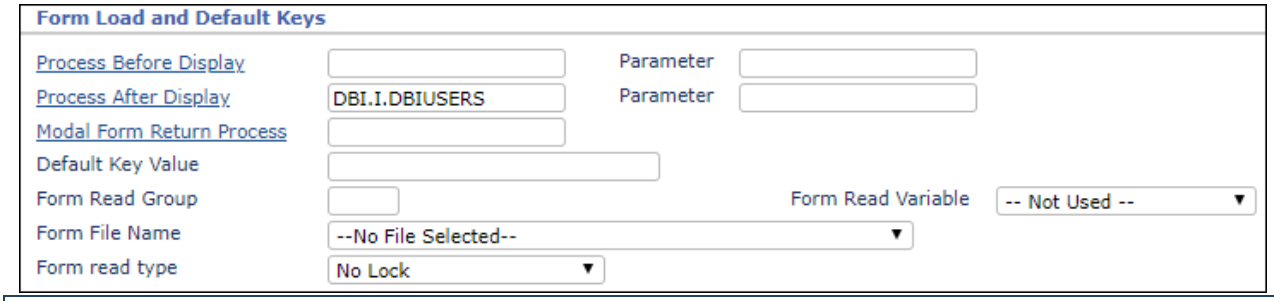
Include Reports in Tab Index

In previous releases INPUT fields in an On-Form Report were assigned a Tab Index of -1 which places



them outside of the normal tab sequence in the browser. This could lead to tabbing to the browser address bar. Checking "Include Reports" will add OFRs to the Tab Index calculations and the Tab Index Properties maintenance form. Any Input fields in the OFR will then be assigned the given Tab Index. For backwards compatibility this is optional.

Section 2:



Prompts

Process Before Display — Is used to define a subroutine name that is to be called before the form is displayed.
This is very useful for pre-form security checking processes, or to modify the form before the end-user views it, or to move to a different form based on business rules.

This will invoke the subroutine with a "BEFORE DISPLAY" event. If SCREENROOT is modified, it should be done so via PROCESS.STACK.

DesignBais calls the BEFORE DISPLAY process twice.

To stop the second call to the BEFORE DISPLAY developers can now set PROCESS.EVENT = "NOREPEAT BEFORE DISPLAY" at the end of their BEFORE DISPLAY processing logic.

Most *Process* slots now have the tag as a clickable hyperlink. Click this link to edit the routine in Code Editor. Refer to the Code Editor section of this manual for more details.

Process After Display — Is used to define a subroutine name that is to be called after the form is displayed. Note that you cannot call another form from this slot. The error "*Not a valid subroutine. You may only enter a cataloged program name*" displays if you attempt to enter a *Filename_Formname* process. Use the *Process Before Display* event if you need to call a different form based on business rules.

The PROCESS.EVENT value for this process is "AFTER DISPLAY".

If a derived key has been supplied this process will be called a second time with a process event of "DERIVED KEY"

Parameter — The parameter passed to the After Display event. The variable filled is PROCESS.PARAMETER.

Modal Form Return Process — Is used to define the name of a subroutine to be called when the form gains focus after a modal (or layered) form is closed. Note that the Modal Form Return Process must be defined on the form calling the modal NOT on the modal form that is closing.

The PROCESS.EVENT value for this process is "MODAL RETURN"

Default Key Value — The default key value is used to define a single record key for the form. The default key can be either a string value, or, if prefixed with 'V:' will use a DBWORK variable. The field name entered after the 'V:' must be a valid field name defined in the Field Properties of the form file.

This field is used in conjunction with the following fields:

- Form Read Group
- Form Read Variable
- Form File Name
- Form Read Type

Form Read Group — A number between 1 and 99 that indicates which read group number will be assigned to the read for a derived key.

Form File Name — This field is used to determine which file the record is read from when a Default Key Value is used.

Form Read Variable

DesignBais allows the developer to assign the read record to any one of one hundred read variables, being:

DBRECORD
DBOTHER.RECORD(1) – DBOTHER.RECORD(99)

When defining a derived key you must assign a read variable. Usually this would be DBRECORD.

Form Read Type

When using a Derived Key the read type specifies whether the record will be locked when read.
Note that there is no explicit Process After Read slot when using the Form Read. Use the Process After Display event for after read processing as the read has been completed prior to this event.

Section 3:

Prompts

Include Header Form

This field allows the developer to identify a header form to be included at runtime. The nominated form will be inserted at the top of the main form at runtime. If the nominated form has a Header or Footer Form of its own these will be ignored.

The nominated form is not included at design time, but is in test mode and at run time.

You must ensure that section names in the header form do not conflict with other sections on the main form particularly when using collapsing sections.

A header form may have a height of zero. This allows for the header form to be an overlay, triggered by a Mouseover event. Refer to the header Form Overlay section of this manual.

The width of elements on the header form will influence the display of the main form. Fields on the header form that are wider than the main form will cause a scrollbar to appear in some cases. This behaviour can be averted by the use of a column span value of “100%” on text and output field elements on the header form. The resulting HTML will then include “right:0px” which causes the header form elements to span from their starting column position to the right hand margin of the form. Note that use of “100%” still allows an element to commence a a column position greater then 0.

There are three events supported for all header forms. These events will be run before their partner events in the main form.

Before Display.

If a before display event is defined in the header form the “BEFORE DISPLAY HEADER” event will be invoked.

It is important to note that the before display event should be limited to defining features of a form before display. It can also be used to invoke another form via PROCESS.STACK.

In the event that another form is invoked via PROCESS.STACK, the before display event will be triggered a second time.

This will be invoked before the “BEFORE DISPLAY” event on the invoked form.

After Display

If an after display event is defined in the header form the “AFTER DISPLAY HEADER” event will be invoked.

This is useful to set common variable structures for a header form or the form being invoked.

This will be invoked before the “AFTER DISPLAY” event on the invoked form.

Modal Return

If the modal return event is defined in the header form the “MODAL RETURN HEADER” event will be invoked every time a sub-form, search form or layered form is closed. This is very useful if you wish to modify something on the header form when a modal is closed.

This will be invoked before the “MODAL RETURN” in the current form.

Removing the Header Form

Before removing the header or footer form from a form developers must check that there are no references to buttons on either the header or footer form. For example the *Update From Button* list includes buttons from the header and footer form. If these have been selected and then, at a later date the header or footer form is removed, the form may reference button names that are no longer available. Buttons may also be referenced in the Control+Enter Button Function. These correspond to form attributes:

<72> DBIF.WRITE.FROM.BUTTON – update from button
<135> DBIF.FIELD.CONTROLKEY.LIST – the Control+Enter Button Function

Include Footer Form

This field allows the developer to identify a footer form to be included at runtime. The nominated form will be appended to the bottem of the main form at runtime. If the nominated form has a Header or Footer Form of its own these will be ignored.

The nominated form is not included at design time, but is in test mode and at run time.

You must ensure that section names in the footer form does not conflict with other sections on the main form particularly when using collapsing sections.

Background Color

This field can be used to assign a background color to a form. This field should be used only when you require a form to have a different background color than provided by the style group.

Top Menu

Will determine the Top Menu that is associated with the form. If left not selected, then the form will inherit the menu from one of its parents. Every DesignBais form can be linked to a menu if required. This provides complete flexibility in the design of the user interface.

Side Menu

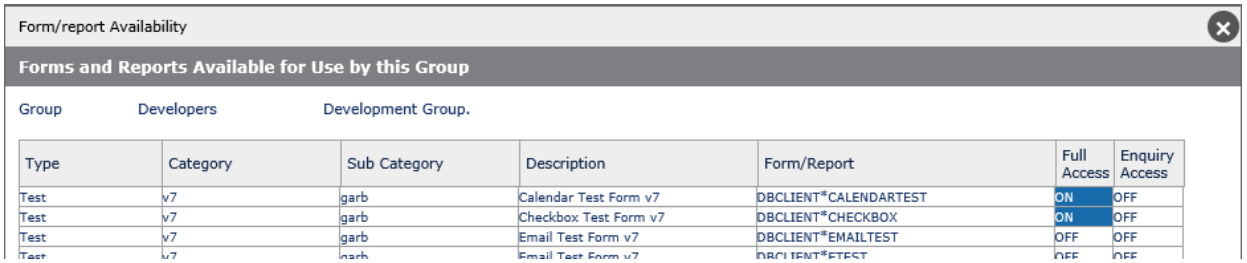
Assigns a side menu to the form. Assign a Hidden Side Menu to eliminate the space allocated for side navigation.

Type

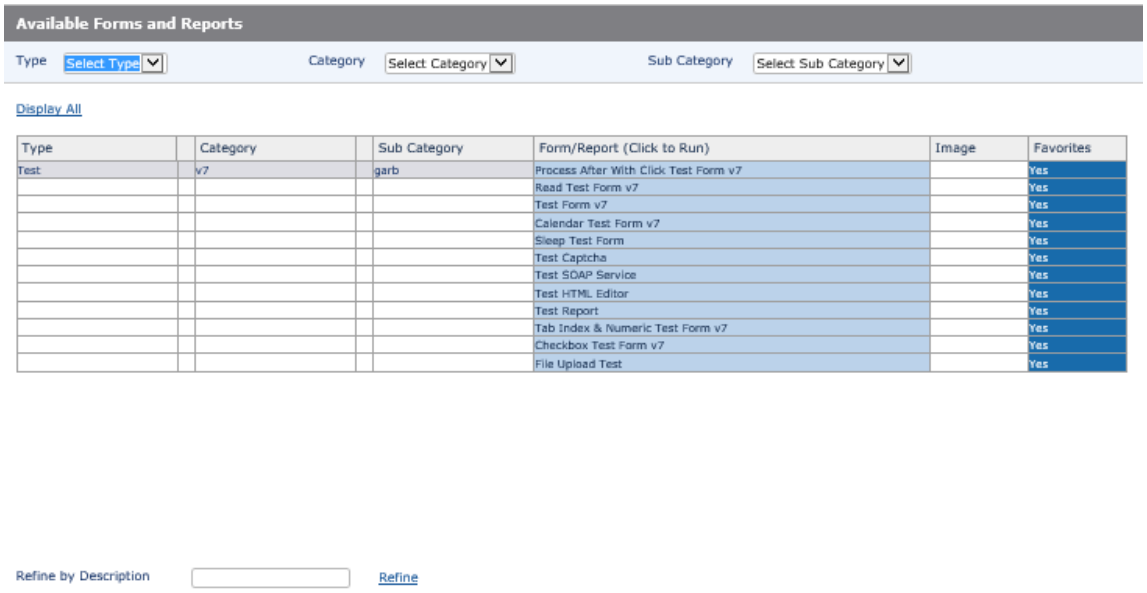
Click this searchlabel to display the maintenance form for Form Types and Categories. Assigning a form to Type, Category and Sub Category is the first action required if you wish to implement Favorites (Favourites).

DesignBais incorporates an On-form Report style menu structure that allows the end-user to select their favorites (favourites) from the menus in order to create a favorites list. The Type, Category and Sub Category entities can be entered for each form or report. Once you have nominated one or more of these three entities for a form or report the description given to the form or the report will be added to this menu's option list.

Use the **Access Control** button option in User Groups to grant users access to forms and reports.



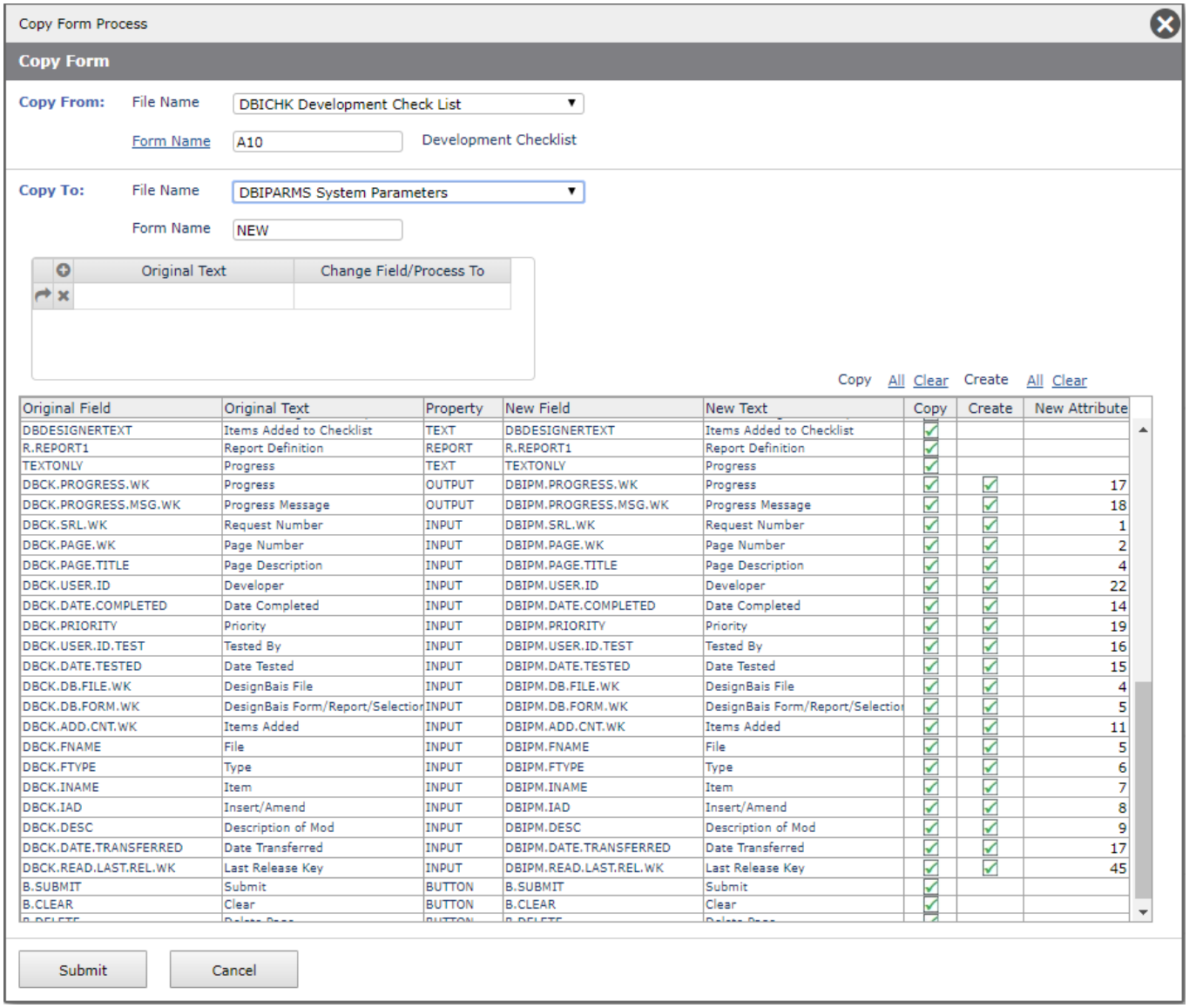
For a form or report to appear in the Favorites menu it is necessary to assign it to a Type, and/or Category and/or Sub Category, to assign access via the Access Control form, and then to click the Favorites column in the "Display All" option of the report displayed when Favorites (on the top menu) is clicked.



The above form allows selection of **Type, Category**, **Sub Category** and the display is filtered accordingly. The **Image** field is the name of the file containing the image of a form. The Image file name feature is not used.

Buttons

Designer	Will update the DBIFORMS record *Filename* Form Name* and invoke the forms designer. (A button is at the bottom of the form and a Designer link at the top.)
Clear	Will clear the form and discard any changes.
Delete	Will delete the form from the DBIFORMS file.
Submit	Will update the DBIFORMS record *Filename* Form Name* but does not invoke the forms designer.
Copy Form	Will copy the current DBIFORMS item. (Link is at top of form only.) Developers can create templates with a standard look and feel, then copy and customize them as required. The copy process allows the developer to change the target file for the form, and to define text changes which will be made in the copy. (Caution: Text changes are global and will affect labels, field names, and program and form references.) From Release 7/8 the Form Copy process has been enhanced to allow for fields on the form being copied to be created as Field Properties on the target file. The Equate Name is changed to that of the target file. The Field Attribute value can be amended prior to the copy. You can also select which fields are to be copied allowing for a new form with just a sub-set of the fields on the source form.



Report — Will produce a report of the fields that appear on the Form. (Link is at top of form only.) See example below.

Form Definition fo DBIFORMS — Printed 21 JUN 2016 15:44
Form D10

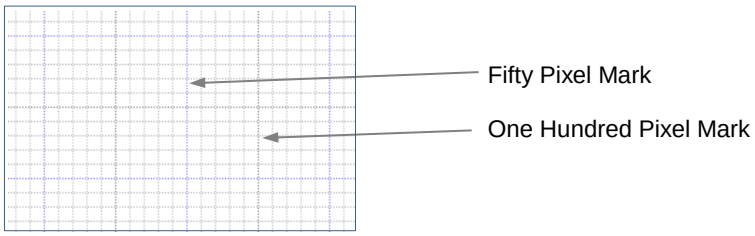
File Name	Field Name	File Variable	Section	Col Span	Row Span	Field Type	Attr	M\	Leng	Display Class	Processes
DBIFORMS	DBIF.FORM.WIDTH	DBRECORD		40	18	INPUT	2	N	4	Input	
DBIFORMS	DBIF.FORM.DEPTH	DBRECORD		40	18	INPUT	3	N	4	Input	
DBIFORMS	DBIF.STYLE.GROUP	DBRECORD		45	18	INPUT	115	N	4	Input	Lookup File : DBISTYLEGROUP
DBIFORMS	DBIF.W3C	DBRECORD	Hidden	21	18	CHECK	143	N	10	Input	
DBIFORMS	DBIF.W3C.PDA	DBRECORD	PDA	76	18	INPUT	178	N	10	Input	
DBIFORMS	DBIF.PRESERVE.COMMON	DBRECORD	Main	21	18	CHECK	5	N	1	Input	
DBIFORMS	DBIF.FORM.MODAL	DBRECORD	Main	21	18	CHECK	4	N	1	Input	
DBIFORMS	DBIF.SUB.FORM	DBRECORD	Main	21	18	CHECK	144	N	1	Input	
DBIFORMS	DBIF.SCRIPT.INCLUDE	DBRECORD	Ajax	550	58	INPUT	11	Y	20	Input	
DBIFORMS	DBIF.TAB.INDEX	DBRECORD	Main	21	18	CHECK	114	N	1	Input	After Process : DBI.I.DBIFORMS
DBIFORMS	DBIF.ENTER.CONTROL	DBRECORD	Main	196	18	INPUT	164	N	30	Input	Lookup File : V:DBIF.BUTTON.FIELDS.LIST.'
DBIFORMS	DBIF.INCLUDE.DROP	DBRECORD	Main	46	18	CHECK	139	N	5	Input	
DBIFORMS	DBIF.FORM.CENTERED	DBRECORD	Main	46	18	CHECK	174	N	5	Input	
DBIFORMS	DBIF.OVERLAY.REQUIRED	DBRECORD	Main	76	18	CHECK	174	N	10	Input	
DBIFORMS	DBIF.BEFORE.DISPLAY	DBRECORD	Main	140	18	INPUT	175	N	20	Input	After Process : DBI.I.DBIFORMS
DBIFORMS	DBIF.BEFORE.DISPLAY.PAR	DBRECORD	Main	140	18	INPUT	175	N	30	Input	
DBIFORMS	DBIF.PROCESS.DURING.LC	DBRECORD		140	18	INPUT	6	N	20	Input	After Process : DBI.I.DBIFORMS
DBIFORMS	DBIF.PARAMETER.DURING.	DBRECORD		140	18	INPUT	7	N	20	Input	
DBIFORMS	DBIF.MODAL.FORM.RETURI	DBRECORD	Main	140	18	INPUT	63	N	20	Input	After Process : DBI.I.DBIFORMS
DBIFORMS	DBIF.KEY.DERIVED	DBRECORD		140	18	INPUT	145	N	20	Input	
DBIFORMS	DBIF.FORM.READGROUP	DBRECORD		35	18	INPUT	146	N	5	Input	
DBIFORMS	DBIF.FORM.READVAR	DBRECORD		140	18	INPUT	147	N	20	Input	
DBIFORMS	DBIF.FORM.FILEVAR	DBRECORD		140	18	INPUT	148	N	20	Input	Lookup File : DBIFILES After Process : DBI.I.DBIFORMS FILE.CHECK
DBIFORMS	DBIF.FORM.READTYPE	DBRECORD		35	18	INPUT	149	N	5	Input	
DBIFORMS	DBIF.FORM.INCLUDE.HEAD	DBRECORD	Main	200	18	INPUT	134	N	20	Input	After Process : DBI.I.DBIFORMS
DBIFORMS	DBIF.FORM.INCLUDE.FOOT	DBRECORD	Main	200	18	INPUT	134	N	20	Input	
DBIFORMS	DBIF.FORM.BACKCOLOR	DBRECORD		140	18	INPUT	113	N	20	Input	
DBIFORMS	DBIF.TOP.MENU	DBRECORD		140	18	INPUT	65	N	20	Input	Lookup File : DBIMENUS,1 After Process : DBI.I.DBIFORMS
DBIFORMS	DBIF.SIDE.MENU	DBRECORD		140	18	INPUT	66	N	20	Input	Lookup File : DBIMENUS,2 After Process : DBI.I.DBIFORMS
DBIFORMS	DBIF.FORM.TYPE	DBRECORD	Main	136	18	INPUT	186	N	20	Input	Lookup File : V:DBIF.FORM.TYPE.LIST.WK

Gn – Read Group Sn – Read Step Dx – Delimeter R0 – DBRECORD Rn – DBOTHER.RECORD(n) Tn – Read Type
DBIFORMS_R10 Form Definition

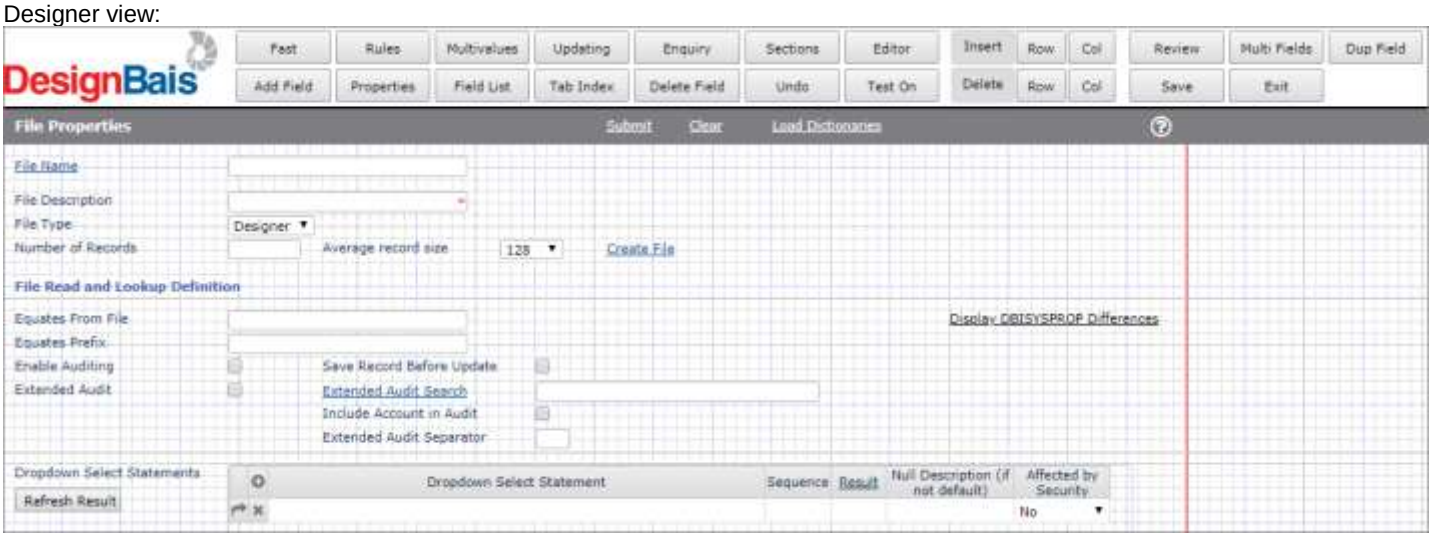
Page 3 of 4

Designer Usage

The first thing you are presented with in the forms designer is a blank canvas with a grid background. The grid has a size of ten by ten pixels. There is a blue line indicating the fifty pixel and a dark grey indicating the one hundred pixel positions.



Designer view:



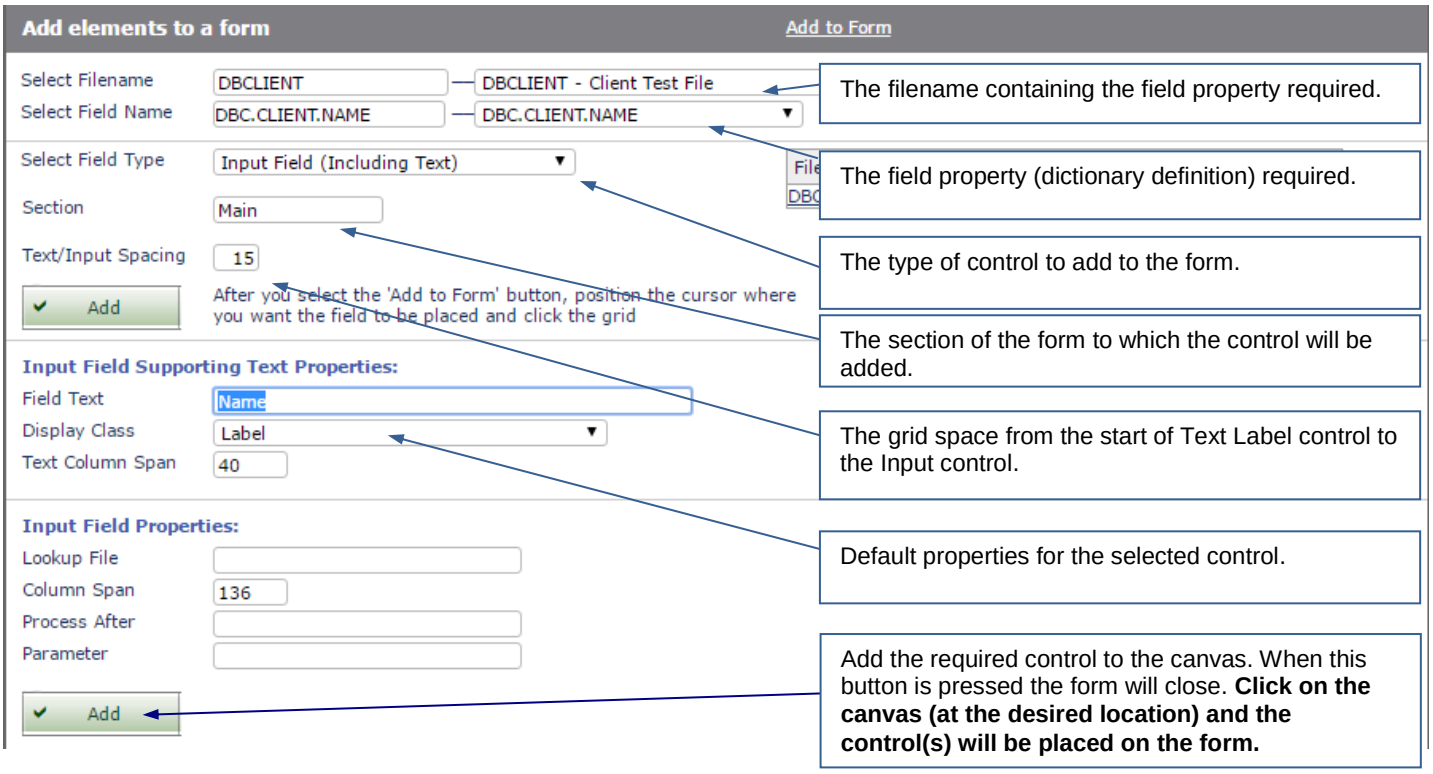
The designer provides all of the options at the top.

Adding a control to the form

Add Field menu option

To add a control to the form, click the **Add Field** button in the top menu.

The following form is displayed:



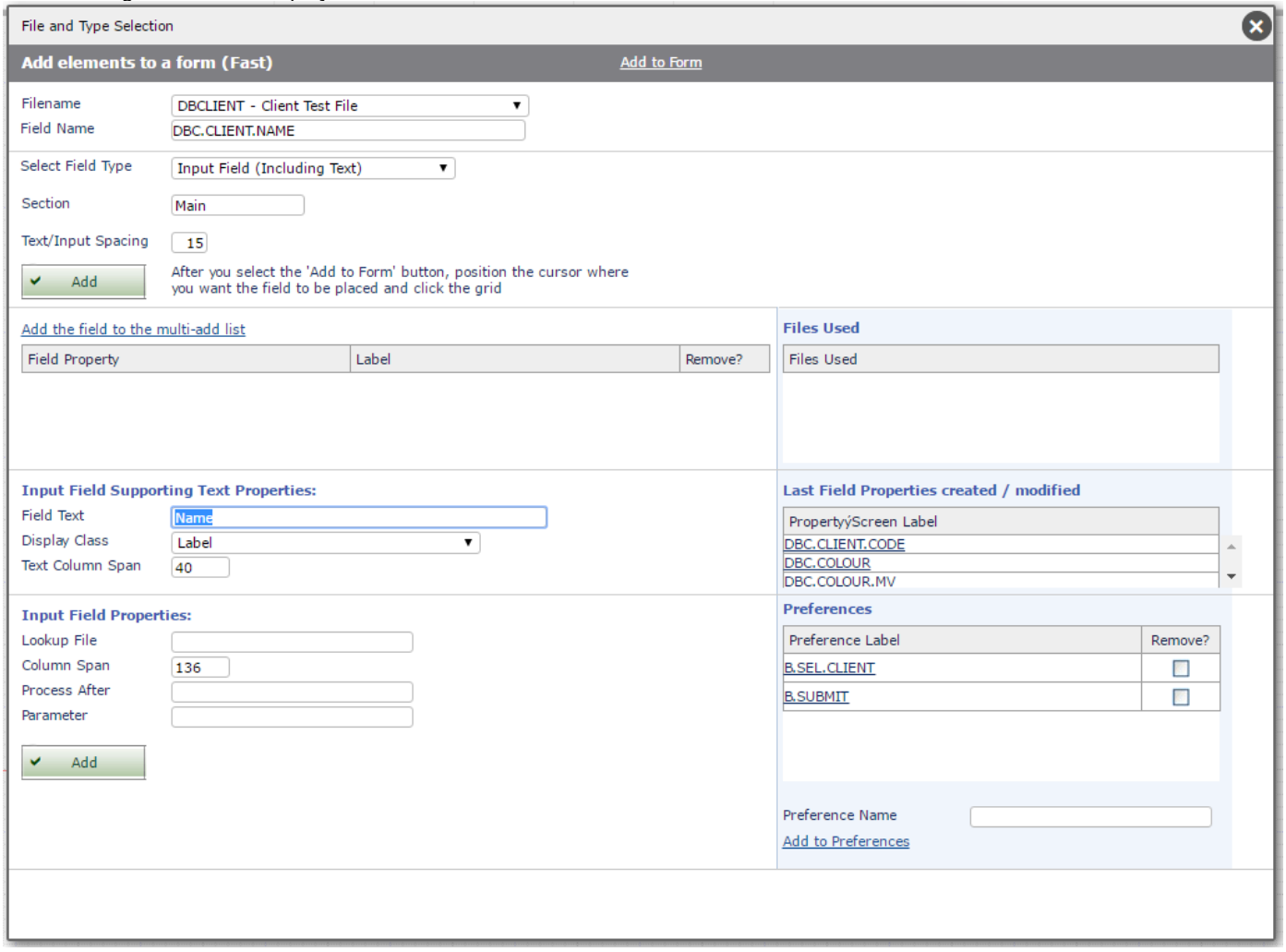
Note: for any file all fields on a form that refer to the same attribute of the file must have a validation process in order to ensure there is a server hit after every change to any of these fields.

Adding a control to the form (FAST)

This provides an alternative Add Field form that provides some time saving features, particularly where files have a large number of dictionaries.

Select **Fast** from the forms designer menu.

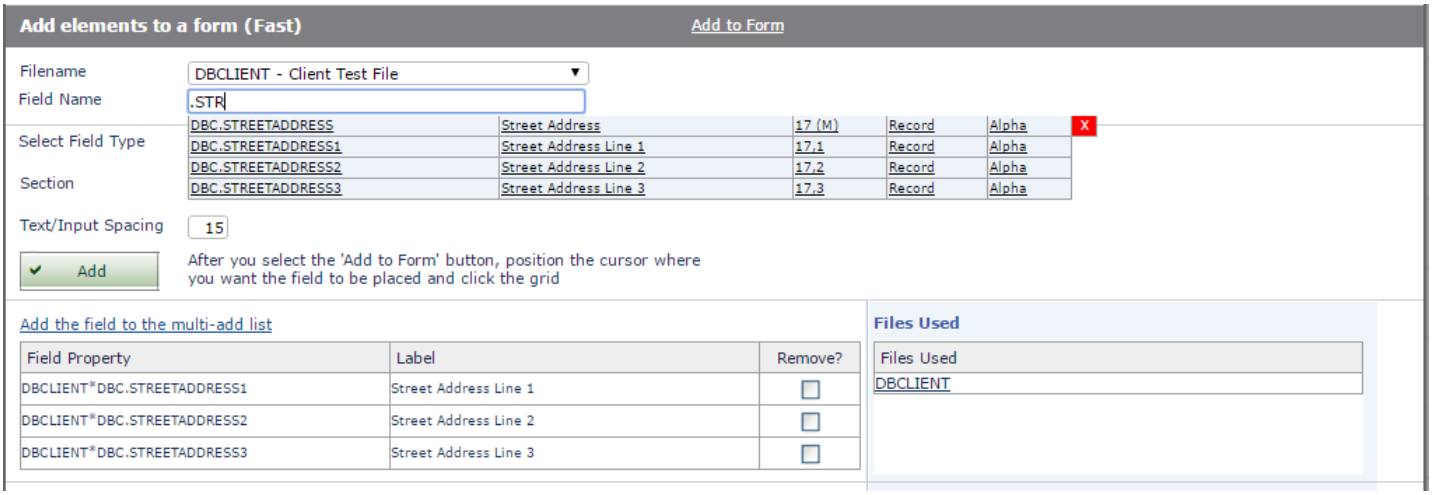
The following form is then displayed.



The Field name prompt in this form is now a keypress field. Every key stroke is transmitted to the database to allow for matching.

Example. In the following example ".str" is entered into the field name. All of the Fields in the file DBCLIENT with a name that contains ".str" is displayed in the dropdown list. Note: a maximum of 40 lines will display.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3



It is now possible to select the required field by clicking on a link in the list.

To close the list without making a selection, select close (x) button.

If system response time is noticeable it is recommended that entry be limited to 3 characters initially. This allows for any auto selections to complete. Once the list has been displayed continue to enter additional search characters if required.

Multiple selections can be made (non MV fields) by holding the Control button down when clicking on an item in the list. This will add the field to the multi-add list.

Multi-add List (Forms Designer Fast button)

The multi-add list option allows the developer to select multiple fields and add them to the form in a single operation.

There are two ways to add fields to the Multi-add list.

Add the field to the multi-add list

This option will add the selected field or preference to the multi-add list.
Any current list of fields being displayed is cleared.

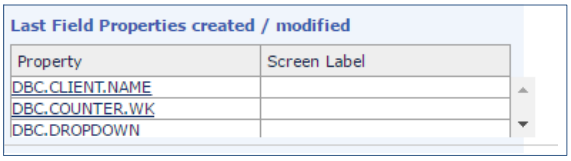
The list of fields will be added when the **Add** button is selected. These fields will be added in a single add operation. All fields will be added row-by-row.

Multi-add is not available for multi-value input and output fields.

Last Field Properties created / modified (Forms Designer Fast button)

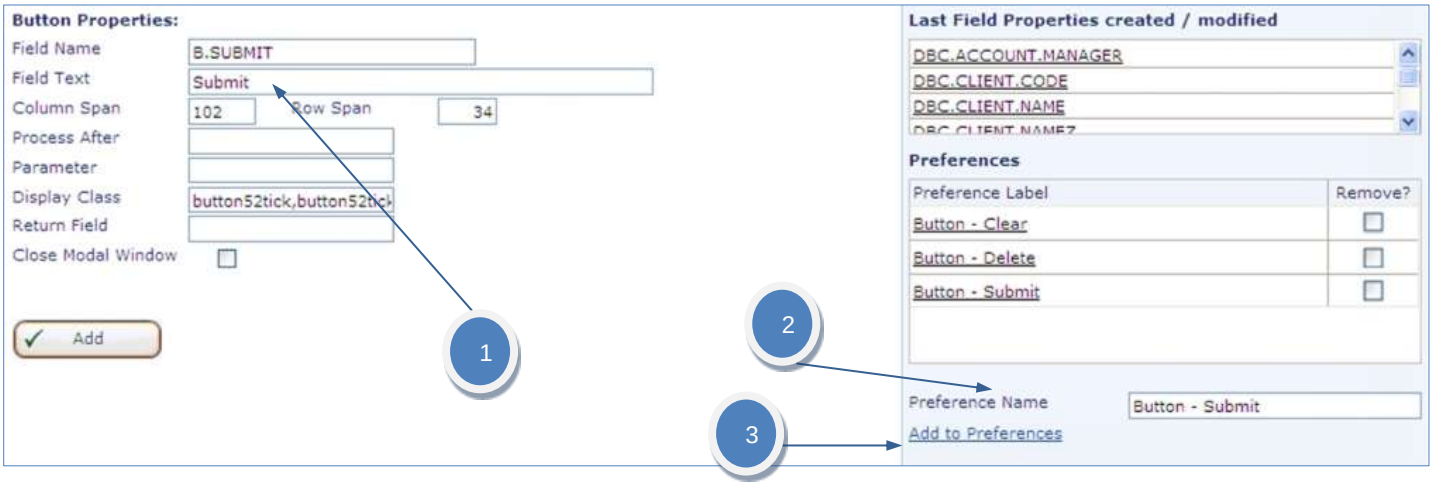
This list displays the last twenty fields created for the selected file. This helps to easily identify new fields (dictionaries) that a developer has created for a file.

Click on the field listed to add it to a form.



Preferences (Forms Designer Fast button)

Preferences allow the developer to store controls that are used regularly so that they may be recalled and added to forms at a later date. This simplifies the repetitive entry of controls on multiple forms. This feature is most useful for form elements that are not directly linked to a field on a file, elements such as buttons (eg Submit, Delete and Cancel) or reports, that are typically added to multiple forms.



Steps for creating a Preference

1. Enter the field properties as you would if the control was being added to a form.
2. Enter the name of the preference. The preference list is sorted, so it is best to prefix the list with a type of control. In the above example “Button” is used.
3. Select the Add to Preferences button and the preference will be added to the preference list.

The preferences are stored against the user record and may be re-used by selecting the required preference from the Preferences list.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Input Field (Including Text)

Places both the Field Property and its description (Screen Label from Field Properties, may be extracted from dictionary items definitions) on the form. The input field is selected (colored blue as seen below). This action creates two controls on the form, the text label and the input field. Both have different property requirements and can be selected and moved individually. If the field being loaded is multi-valued, a text area entry field will be loaded. This will allow for MultiValue entry where a grid-style entry is not required. A value mark is entered into the input string when the user hits the [Enter] key.

DesignBais processes text area fields as follows:

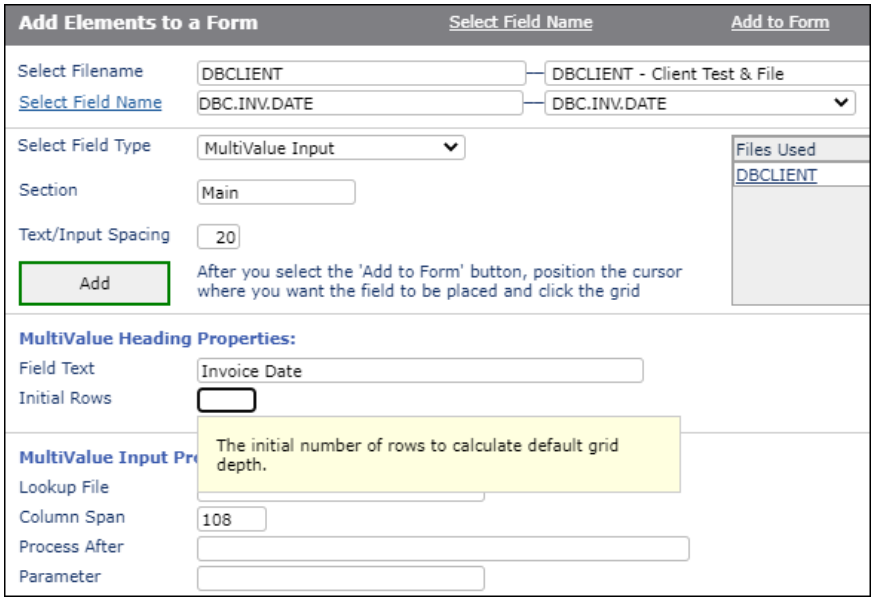
Field Multivalued	Field Subvalued	Group Name	DesignBais action
Yes	No	No	Text will be multivalued with no line breaks.
Yes	Yes	No	Text will be placed in a single value, with line breaks flagged by subvalue marks.
Yes	Yes	Yes	This is the setup required for a text field in a grid when you want line wrapping. Text will be multivalued corresponding to row and subvalue marks will flag line breaks.
Yes	No	Yes	This is the normal setting for a text field in a grid. Text will be multivalued corresponding to row number. Text will not wrap within the grid entry field.
No	Yes	No	This setup defines data that resides in a designated subvalue position of a field. Typically this may occur where several related fields are stored in a single value of a defined attribute.

Input Field

Places an input field on the form for the selected Field Property (no text label).

MultiValue Input

Places a MultiValue input grid on the form at the required location. The rowspan of each row in a grid is set to 18 unless one or more of the fields in the grid are flagged as a sub-valued field. The sub-value flag is the method to allow a field to be used for multi-line text entry. In this case the grid rowspan is set to the rowspan of the sub-value flagged field that has the greatest rowspan.

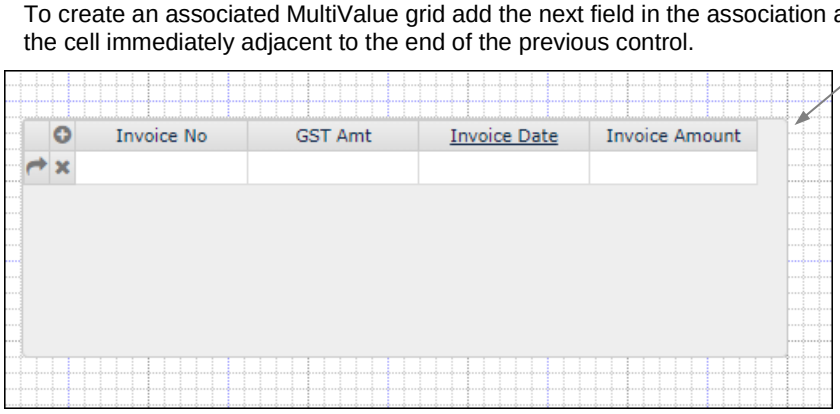


Initial Rows

Enter the number of rows that you want to see in the grid. This is rows not pixels.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

To create an associated MultiValue grid add the next field in the association and click on the canvas in the cell immediately adjacent to the end of the previous control.



The placement of fields in a multivalue grid can be physically changed by changing their tab index, or by changing their column number:

- If Input Fields Use Tab Index is **on** then use Tab Index.
- If Input Fields Use Tab Index is **off** then use Column Number.

This is most easily done by using the Field List form within Forms Designer as this displays simultaneously the column number and tab index values for all fields in a grid. Note that if all fields within a grid have the same Tab Index value, which is valid, then Column Number can be used to re-order the grid columns.

Note that tab index values for a particular multivalue grid must be contiguous. If a field that is not in the grid has a tab index value that lies within the range of tab indices of the fields that are in the grid then the tab sequence will not be followed after a change event. If a field in the grid, other than the last column of the grid, is amended then focus will not flow to the next field in that row, but will jump to the next row of the grid.

The Upgrade Routines (within *Upgrade/Migration Tools* on the Development Tools side menu) option *11 Fix Form MV Grid Tab Sequence to be the same for all fields on the grid* will report forms where the multivalue grid tab sequence is not sequential. These forms can be fixed using the update option within this tool.

The width of a multivalue grid is calculated by summing the colspan of each field in the grid and then subtracting the the number of pixels required for the grid controls:

- Append / Delete a grid row (the “+” and “X” icons shown in the example above) 16 pixels
- Insert a grid row (the curved arrow icon shown in the example above) 16 pixels
- The vertical scrollbar 18 pixels

If any of the Append, Delete, and Insert row functions are marked as “Not allowed” then the grid width is not adjusted for this function.

For a grid containing 3 fields with colspans of 120, 280, and 100 and with all functions allowed the calculation is: (120 + 280 + 100) – 16 – 16 -18 = 450 pixels. The 450 pixels are allotted pro rata to the 3 columns. So the actual colspans will be 108, 252 and 90 respectively.

In order to provide backward compatibility, however, the width of a grid containing just 1 field (column) will not be reduced by the 50 pixels shown above (16, 16 and 18).

Border Width may be specified in the Style Group “MV Border Class”. All border elements in the “Additional Style Properties” are examined and the four border widths calculated are subtracted from the available space for the MultiValue data grid.

There is a “Suppress Scrollbar” option on a MultiValue Grid in Forms Designer. Setting it to “Yes” works the same as DBMVPROP<11>=1. If either are set then the 18px space allowed for the scrollbar is available to the data columns of the grid.

	Invoice Number	Description
	2	Iron
	3	Linen
	4	Zinc
	1	Aluminium

Line #	Common Variable Content
1	DBKEYS(2)<1,1>=2
2	DBKEYS(2)<2,1>=3
3	DBKEYS(2)<3,1>=4
4	DBKEYS(2)<4,1>=1
5	DBOTHER.RECORD(2)<1,1>=Iron
6	DBOTHER.RECORD(2)<1,2>=17741
7	DBOTHER.RECORD(2)<1,3>=BOBG
8	DBOTHER.RECORD(2)<2,1>=Linen
9	DBOTHER.RECORD(2)<2,2>=17378
10	DBOTHER.RECORD(2)<2,3>=bobg
11	DBOTHER.RECORD(2)<3,1>=Zinc
12	DBOTHER.RECORD(2)<3,2>=17367
13	DBOTHER.RECORD(2)<3,3>=bobg
14	DBOTHER.RECORD(2)<4,1>=Aluminium
15	DBOTHER.RECORD(2)<4,2>=17741
16	DBOTHER.RECORD(2)<4,3>=BOBG

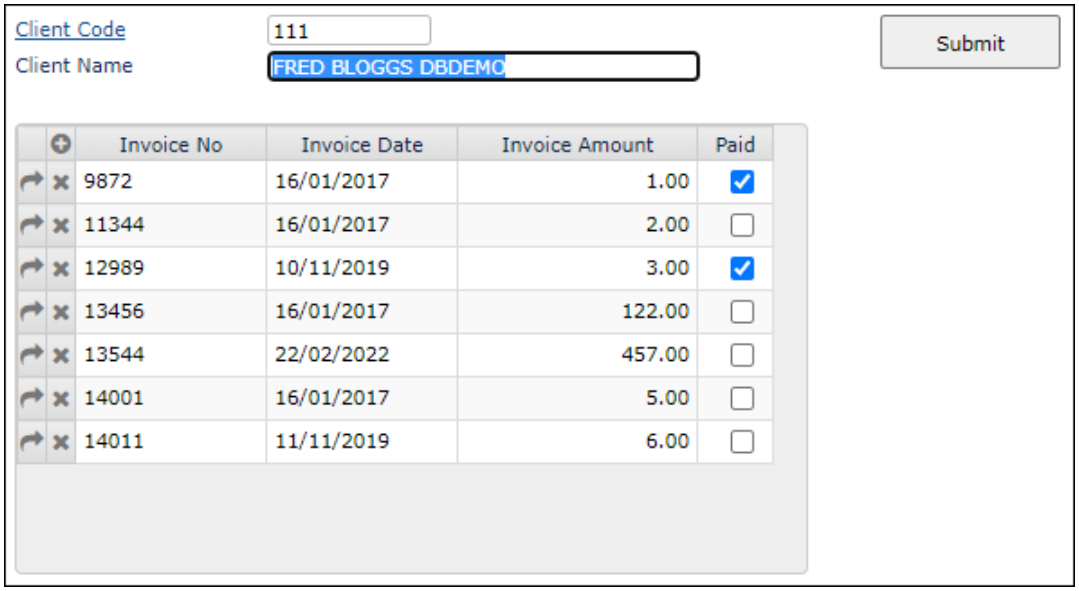
If any field in a Read Group is a multivalue element on the form then the read is treated as a multivalue read type. This means that each row of the grid array performs a read of the record and lowers the attribute marks to value marks in the read variable.

In this example the Invoice number field performs a read on another file where each record contains 3 attributes, into read variable DBOTHER.RECORD(2).

The contents of DBOTHER.RECORD(2) are shown here. Note that each record occupies an attribute. Each attribute of a record occupies a multivalue position within DBOTHER.RECORD(2).

After release 8.6.0.1 there is an option to display a check box in a multivalue grid.

Click the *Display Check box* option and enter the value corresponding to *True* and *False*. The field used to hold the check box will be associated with the other fields in the grid and should therefore have the same Group Name.



MultiValue Output

Places a MultiValue grid on the form that cannot be edited. This is useful to display translated descriptions or calculated values.

Output Field (Including Text)

Places an output-only field on the form including the supporting screen label from the Field Property.

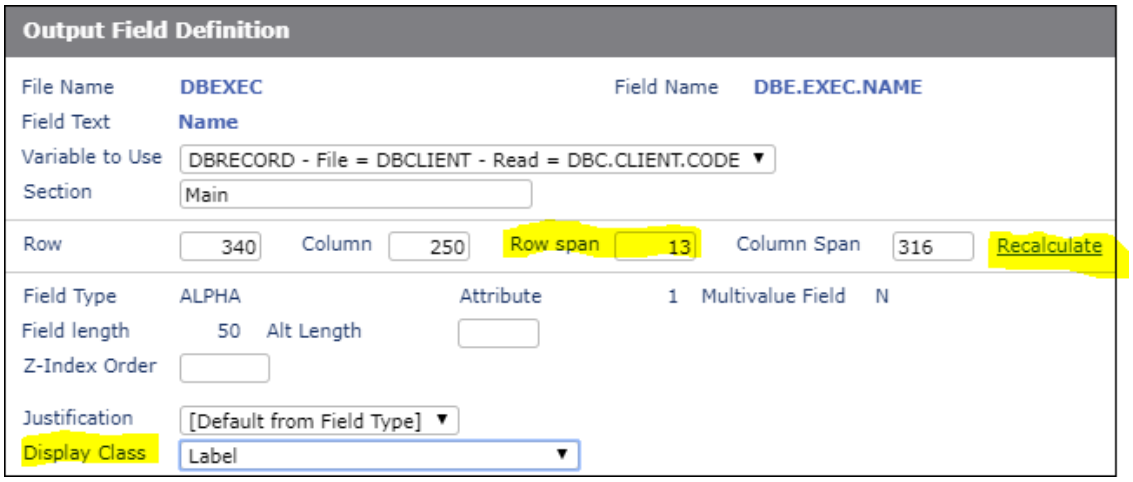
These fields are useful for displaying translated fields or calculations. If the field is Multivalued, a new line will occur for each value mark encountered in the string.

The "Encode HTML" property should be set to "Yes" to prevent Cross-Site Scripting (XSS) attacks unless the application specifically requires HTML to be enabled for the field.

Output Field

Places an output-only field on the form without the supporting text label from the Field Property.
The "Encode HTML" property should be set to "Yes" to prevent Cross-Site Scripting (XSS) attacks unless the application specifically requires HTML to be enabled for the field.

From Release 8.4.1.1 there is a *Recalculate* button on the properties form which when clicked will recalculate the row span based on the display class.



Text From Field

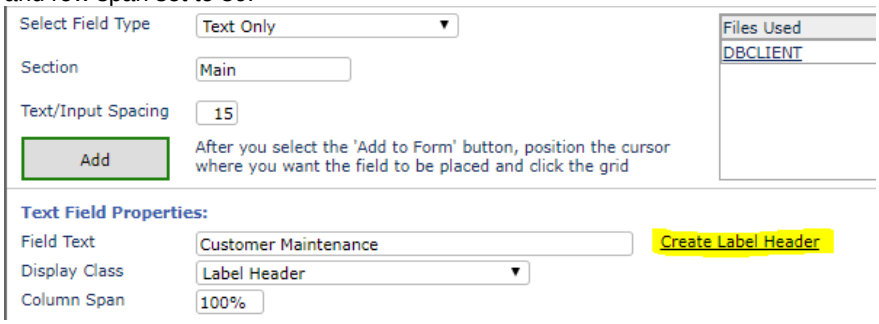
Places only the screen label from a Field Property on the form.
The "Encode HTML" property should be set to "Yes" to prevent Cross-Site Scripting (XSS) attacks unless the application specifically requires HTML to be enabled for the field.

Text Only

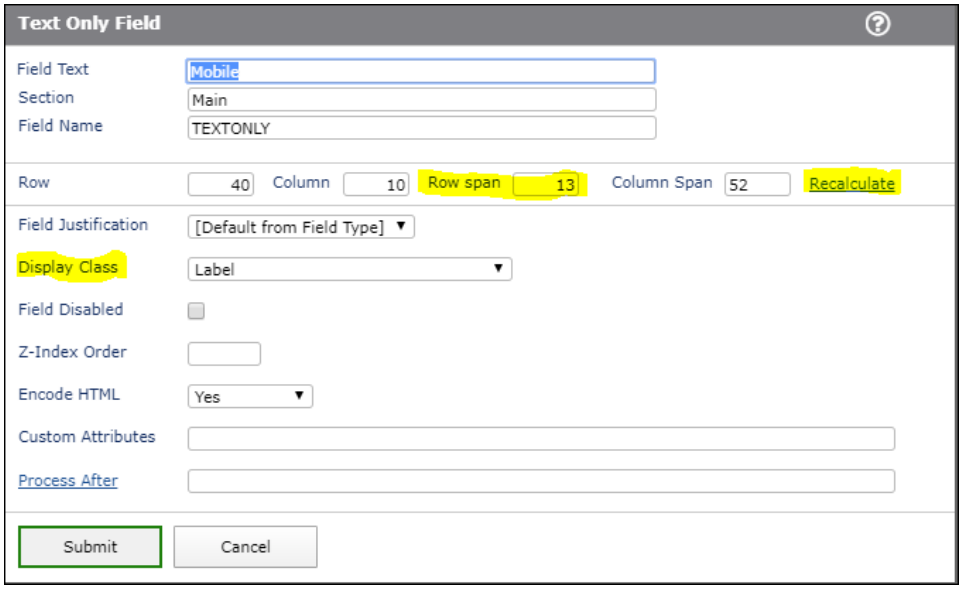
Places a field on the form as a text label. This is commonly used for headings or supporting text on a form.

The "Encode HTML" property should be set to "Yes" to prevent Cross-Site Scripting (XSS) attacks unless the application specifically requires HTML to be enabled for the field.

The *Create Label Header* button allows you to create Label Header text field with column span set to 100% and row span set to 30.



From Release 8.4.1.1 there is a *Recalculate* button on the properties form which when clicked will recalculate the row span based on the display class.



Text only fields receive a BUTTON event for a Text Only field with a Process After.

The developer must give the textonly field a unique name so that the field clicked can be identified in the application code. The default field names of DBDESIGNERTEXT and TEXTONLY will not be useful in this situatiion.

The inline style "pointer-events" will be set to "none" for disabled TEXT fields and to "auto" for enabled TEXT fields.

Button

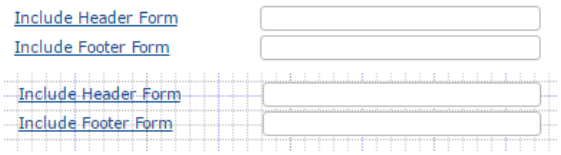
Places a button on the form. A button may be used to load another form, execute a subroutine or close a modal form. There are two styles of buttons that come with standard DesignBais, but new styles can easily be created to suit an application.

When adding a button to a form the name will default to ‘B.BUTTONn’ where ‘n’ is a number 1 greater than the number of controls already on the form.

The style is specified in the field named Display Style:

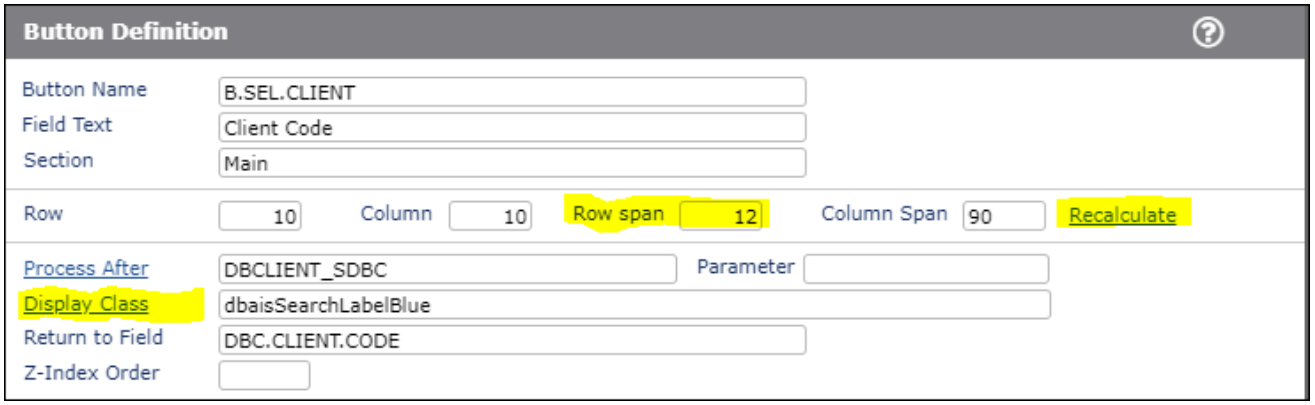
Default Style (Display Style is null)

Hyperlink Style named: dbaisSearchLabel



From Release 8.4.1.1 the Forms Designer has been enhanced to respond to the display classes within the style group that has been assigned to the form. The developer can now use the *Recalculate* button to recalculate the *Row Span* value based on the *Display Class.*

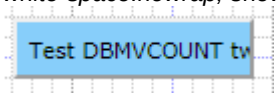
The developer can still amend the row span to any desired value. The *Recalculate* button references the style record for the display class and sets the row span based on the font or the height element (e.g. height: 12px).



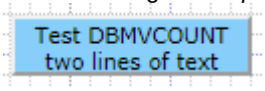
From Release 8.5.1.2 the “white-space:nowrap;” style property has been removed from normal buttons.

The following examples show the effect of this change. Consider a button 30px high (the DesignBais default) with three lines of text:

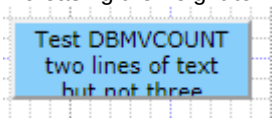
white-space:nowrap; shows text truncated:



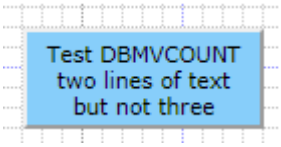
After removing *white-space:nowrap;* the button text is displayed in 2 lines:



Increasing the height to 40px allows the display of part of the third line:



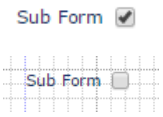
Increasing the height to 50px allows for the display of three lines but a slightly greater increase would create a better look:



Developers may find in some cases that there is too much text for the button width. Up to now this has been hidden as their button height does not allow the extra text to display. This can be corrected in Forms Designer.

Check Box (Including Text)

Places a Check box including the supporting text from the Field Property on the form. This type places two controls on the form, one for the supporting text and one for the Check box. Each control is independent of the other.



Check Box

Places a Check box on the form without the screen label. One way to simulate radio buttons is to associate multiple check boxes with a Group Name. In some cases a check box may require a process after event in order to refresh the browser with the correct value for the field represented by the check box. This can be a “dummy” process in that it is only required to achieve a server hit. The effect of the dummy server hit is to align SCREENREC (DesignBais common variable) with the current values in DBRECORD or DBWORK.

Report

Places an On-form Report on the form. This control allows the developer to create a free-form report on the form. The Report can have click events that invoke subroutine calls. Please see the On-Form reporting

section in the ***Common variable usage and Subroutine interaction*** section of this document for help on constructing On-form Reports.

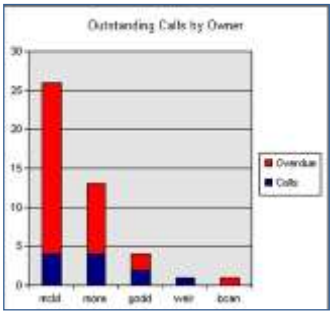
Sale Date	Product	Quantity	Unit Price	Extended Price	Tax	Total (Inc Tax)
30/09/2003	Fine Ballpoint Pen	2	15.30	30.60	3.06	33.66
23/05/2003	Desktop Speakers	3	23.45	70.35	7.04	77.39
	Total	5		100.95	10.10	111.05

The "Encode HTML" property should be set to "Yes" to prevent Cross-Site Scripting (XSS) attacks unless the application specifically requires HTML to be enabled for the field.

Image

Places an image control on the form. The image can be any type that will display in a browser. The default image, shown below, is the DesignBais logo. Use the Properties option to change the default image file. The root path for image files is the images folder in DBWeb virtual directory (c:\db\images). Use relative addressing from that path to point to new files (myapp/logo.jpg points to c:\db\images\myapp\logo.jpg). You can also change the image name in a BASIC subroutine via the DBIMAGESPEC variable. Please refer to the ***Common variable usage and Subroutine interaction*** section of this document for help on DBIMAGESPEC.

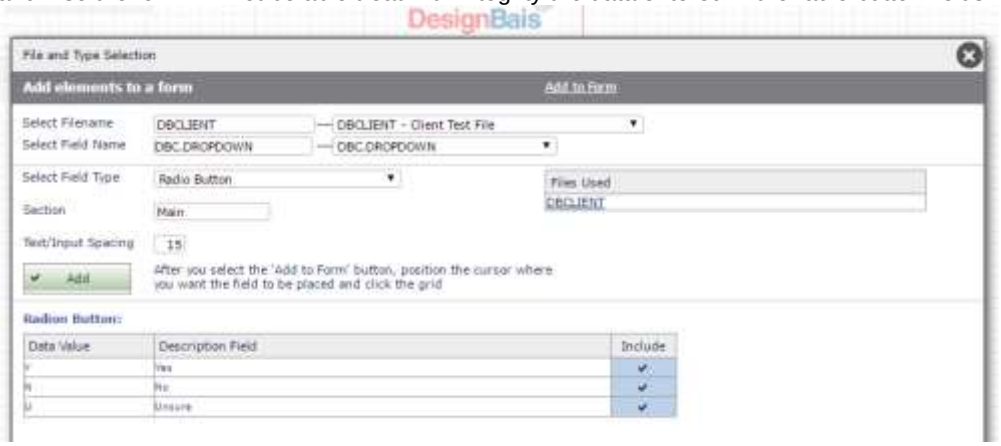
Graph



Not available in Version 7 and later releases. Places a graph control on the form. Please see instructions on setting up a graph in the **Common Variable usage** section of this document. Graphs require the Office Web Component (OWC10) to be installed on the web server. See web component installation instructions for details.

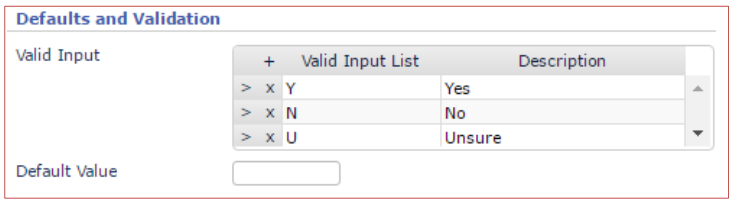
Radio Button

Places a radio button set on the form. The radio button set must be created from a field property that has valid Display and Input values. Note that a radio button set for a field must not be duplicated on a form. There is a warning when the duplicate set is added to the form in Designer but this warning can be ignored and if so the form will not be able treat with integrity the data entered in the radio button fields.



The radio button set will be created as per the Field Property definition for the selected property.

In this example, the field property had the following definition.



This will create the following radio button set in the form designer. The Field Name linked to the radio button must be assigned a value in the AFTER READ event for example. This allows DesignBais to set the display of the active button state. If no value is set all radio buttons will appear as a shaded circle with no balck dot.

Field property names ending with a numeric character will cause problems in DesignBais and should be avoided.

For example the field names DEM.AGREE3 and DEM.AGREE4 when used as radio buttons will be assigned XML Ids of *demagree3* and *demagree4*. But the trailing numeric is used by DesignBais to specify the multivalue position within the form record. One solution is to create fields named *DEM.AGREETHREE* and *DEM.AGREEFOUR*.

Captcha

This will create a field on the form named CAPTCHA. When this field is on the form DesignBais will create a captcha view.

Example:

Refer to details of Captcha usage in the Common Variables and Subroutines section of this manual.

Highcharts Graph

Available in Version 7 and later releases.Places a Highchart graph on the form.

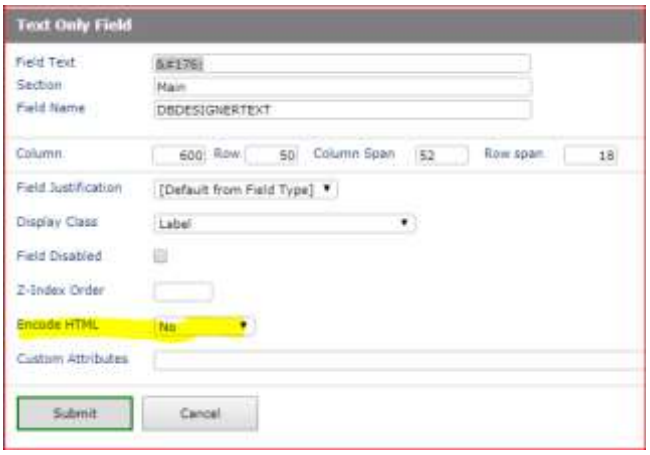
Using HTML encoding

Use HTML encoding to display special characters.

For example:

Degree symbol only	&#176;
Degrees Celsius	&#8451;
Degrees Fahrenheit	&#8457;

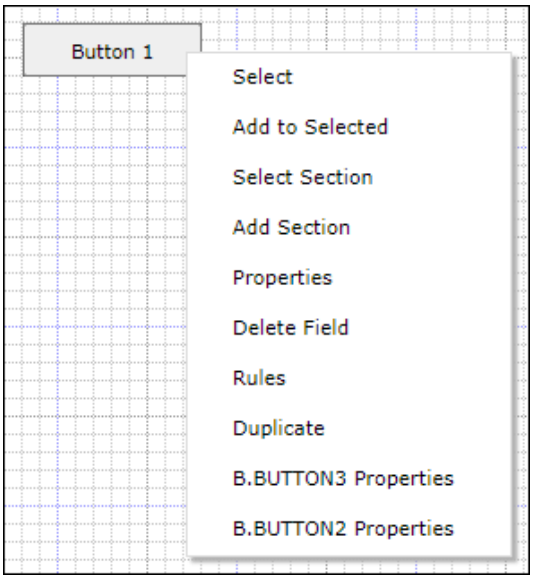
Ensure Encode HTML is set to No for the field.



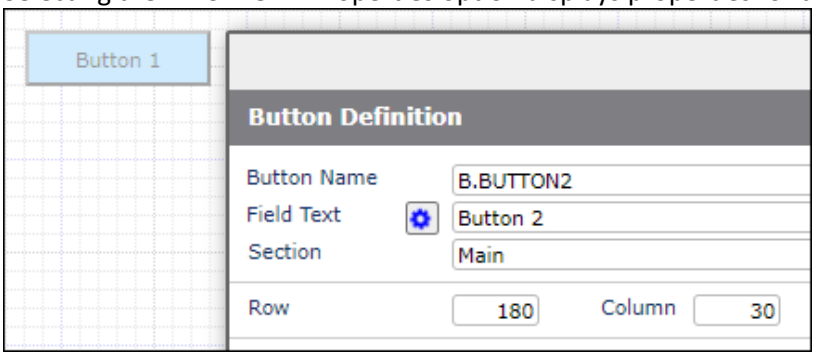
Accessing Form Elements with the same Row and Column Position

Use the context menu feature to access form elements hidden by another element occupying the same row and column position.

Right-click the top element to get a context menu. The options at the bottom of the list allow access to the hidden elements.



Selecting the B.BUTTON2 Properties option displays properties for the hidden button:



The most recently accessed element will remain in view within Designer.

Google Geolocation (Address) Input Field

DesignBais classic forms in Release 8.6.0.1 and above now support the Google Geolocation feature.

To implement an input field using this feature the developer must follow these directions:

- The Google API key must be defined in the *RDPlacesKey* entry in the db.config file.
- There needs to be a script reference as per Global or System Meta Data or in the website custom.js file or perhaps in the website admin/htmlMetaTags.txt item:

  <script type="text/javascript"
  src="https://maps.googleapis.com/maps/api/js?key=GOOGLEKEY&callback=geoInitialize&libraries=places"
  async="" defer=""></script>

  GOOGLEKEY is valid if <RDPlacesKey> is set otherwise the full API key can be used.

- In the AFTER DISPLAY event for the form insert the following:

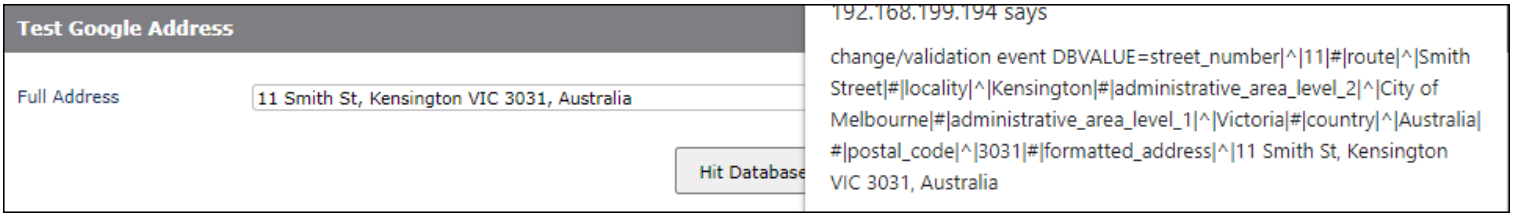
  DBAJAXCMD<-1>='geoInitialize(); '

  Note that the three trailing spaces are required to form the DesignBais javascript termination string.

- In Forms Designer the field requires the Custom Attributes: *geoloc="true" address="".*
- Add the *countryFilter="AU|NZ"* to the Custom Attributes string if addresses are to be limited to particular countries, where the pipe separated string is a list of valid country codes to restrict the address search.
- Alternatively you can use the DBI.G.AJXCMD subroutine in your basic code:

  AJX.FUNC = 'CA'
  AJX.DATA = 'countryFilter="AU|NZ"
  CALL DBI.G.AJXCMD(AJX.FUNC,AJX.DATA)

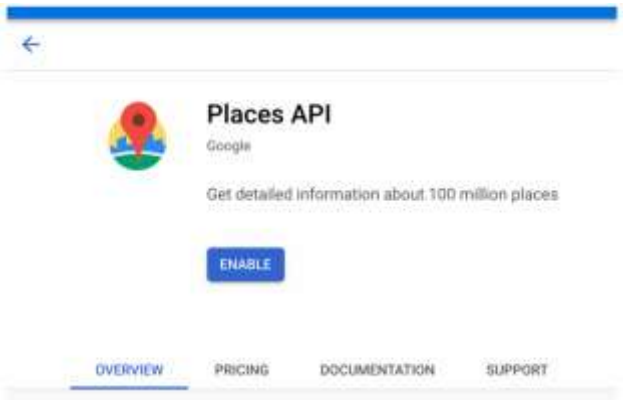
The value returned in DBVALUE will be formatted as shown in the snip below.

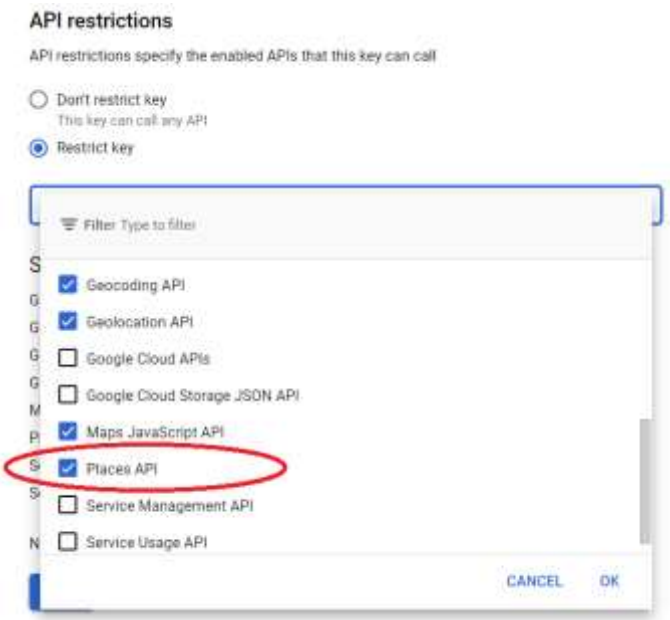


Assuming that the variable *RAW.ADDRESS* contains the entire Google address then the formatted address can be extracted with code such as:

```
FLD.SEP = "formatted_address"
IDX = INDEX(RAW.ADDRESS,FLD.SEP,1)
IF IDX THEN
   FMT.ADDR = OCONV(RAW.ADDRESS [IDX,LEN(RAW.ADDRESS)-IDX+1],'G2|1')
END ELSE
   FMT.ADDR = RAW.ADDRESS
END
DBRECORD<BNK.ADDRESS.FMT> = FMT.ADDR
```

In the browser you must enable the “Places API” and Add “Places API” to API Key:





And you should then see addresses returned in your search field:

Progress Bar

In order to display a progress bar on a form the following method can be used.

Add a work output field to the form. Within the Basic routine associated with the form set the value of this output field:

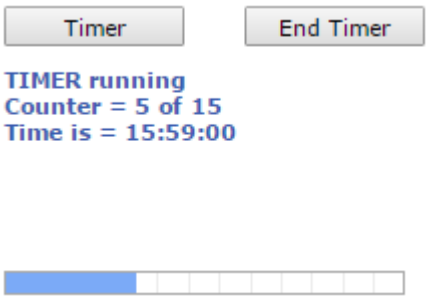
DBWORK<n> = '<progress class="anyClass" value="':PROGRESS.VAL:'" style="width:200px;"></progress>'

where PROGRESS.VAL is set to represent the progress of the function being monitored. In practice this is a number that progressively ranges from 0 to 1.

The use of the **class=”anyClass”** can be used to style the progress bar but can be omitted. The width can be set as required.

In this example DBTIMER is used to control the progress bar:

```
CASE THIS.EVENT = "TIMER"
   GOSUB TIMER.PARA

TIMER.PARA:
   BEGIN CASE
      CASE EVENTSOURCE = “MYTIMER”
         LIM = 20
         DBWORK<DBC.COUNTER.WK> += 1
         IF DBWORK<DBC.COUNTER.WK> > LIM THEN
            NULL; timer has finished
         END ELSE
            DBTIMER<1,1>=7000
            DBTIMER<1,3>="TIMER"
            DBTIMER<1,4>="DB.I.DBCLIENT"       ←the name of your basic subroutine
            DBTIMER<1,5>="MYTIMER"             ←the name of your eventsource
            *
            PROGRESS.VAL = DBWORK<DBC.COUNTER.WK> / LIM '2'
            DBWORK<DBC.NOTE.WK>='<progress class="anyClass" value="':PROGRESS.VAL:'" style="width:200px;"></progress>'
         END
```



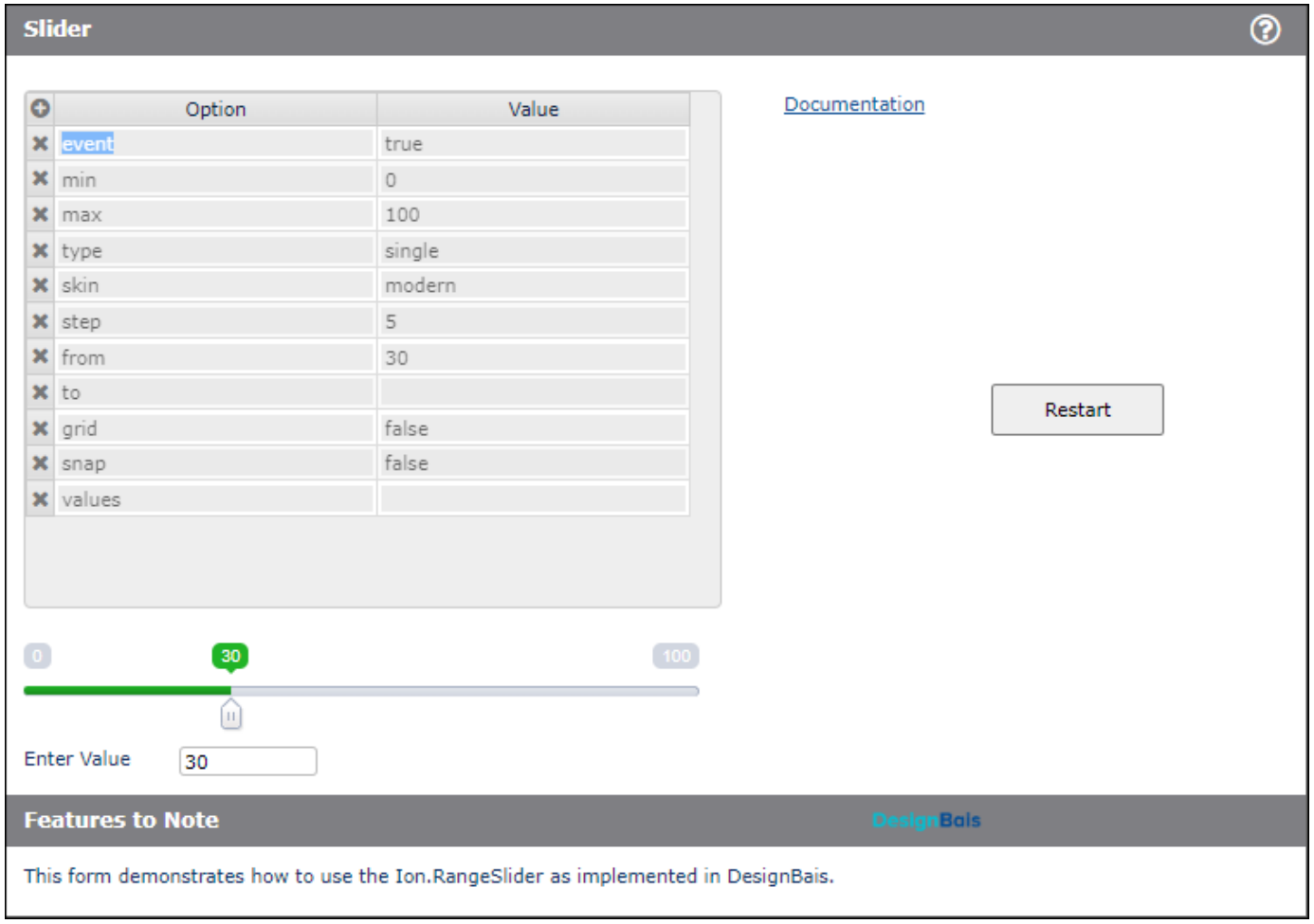
Slider Control

A Slider can be included in both RD and non-RD DesignBais forms.

Refer to the DesignBais Responsive Design Manual section Implementing a Slider if you want to add a slider to a Responsive Design form.

For a non-RD DesignBais form there is a Slider demo form DBDEMO*SLIDER, which is shown below, in the DBINET.DEMO account. The form and the basic code behind it are available by clicking in column 1 of the *Slider* row of the Demo Forms list which can be displayed by selecting the Demo Forms menu option on the top menu of the Development Tools form.

The basic code used to drive the display of the slider is contained in the DBLIB subroutine DBI.I.DEMO which is included in the DesignBais release in account DBINET.DEMO.



The basic code below is extracted from DBI.I.DEMO and is included here to demonstrate how to display the slider from a button click.

The Option and Value grid shown in the snip above are held in work fields DEM.TEMP.FIELD.WK and DEM.TEMP.VALUE.WK. The code below shows how these values are extracted from the work fields and loaded into the array AJX.DATAIN which is then passed into DBI.G.AJXCMD using the 'SL' option.

```
3310        CASE SCREEN.NO = "SLIDER" OR SCREEN.NO = "RDSLIDER"
3311           BEGIN CASE
3312              CASE EVENTSOURCE = "B.SLIDER"
3313                 * Reset displayed value
3314                 VFROM = ''
3315                 VTO = ''
3316                 VSEP = ';'
3317                 VTYP = ''
3318                 * Put all options into the list
3319                 AJX.DATAIN = ''
3320                 AJX.DATAIN<1,-1>  = 'DEM.WORK1.WK'
3321                 JMAX = DCOUNT(DBWORK<DEM.TEMP.FIELD.WK>,VM)
3322                 FOR J=1 TO JMAX
3323                    IF DBWORK<DEM.TEMP.FIELD.WK,J> # '' AND DBWORK<DEM.TEMP.VALUE.WK,J> # '' THEN
3324                       AJX.DATAIN<3,1,-1> = DBWORK<DEM.TEMP.FIELD.WK,J>
3325                       AJX.DATAIN<4,1,-1> = DBWORK<DEM.TEMP.VALUE.WK,J>
3326                       IF DBWORK<DEM.TEMP.FIELD.WK,J> = 'type' THEN
3327                          VTYP = DBWORK<DEM.TEMP.VALUE.WK,J>
3328                       END
3329                       IF DBWORK<DEM.TEMP.FIELD.WK,J> = 'from' THEN
3330                          VFROM = DBWORK<DEM.TEMP.VALUE.WK,J>
3331                       END
3332                       IF DBWORK<DEM.TEMP.FIELD.WK,J> = 'to' THEN
3333                          VTO = DBWORK<DEM.TEMP.VALUE.WK,J>
3334                       END
3335                       IF DBWORK<DEM.TEMP.FIELD.WK,J> = 'from' THEN
3336                          VSEP = DBWORK<DEM.TEMP.VALUE.WK,J>
3337                       END
3338                    END
3339                 NEXT J
3340                 CALL DBI.G.AJXCMD('SL',AJX.DATAIN)
3341                 *
3342                 IF VTYP # 'double' THEN
3343                    DBWORK<DEM.WORK1.WK> = VFROM
3344                 END ELSE
3345                    DBWORK<DEM.WORK1.WK> = VFROM:VSEP:VTO
3346                 END
3347                 * Display Slider
3348                 DBWORK<DEM.WORK3.WK> = 1
```

Slider Panel Control

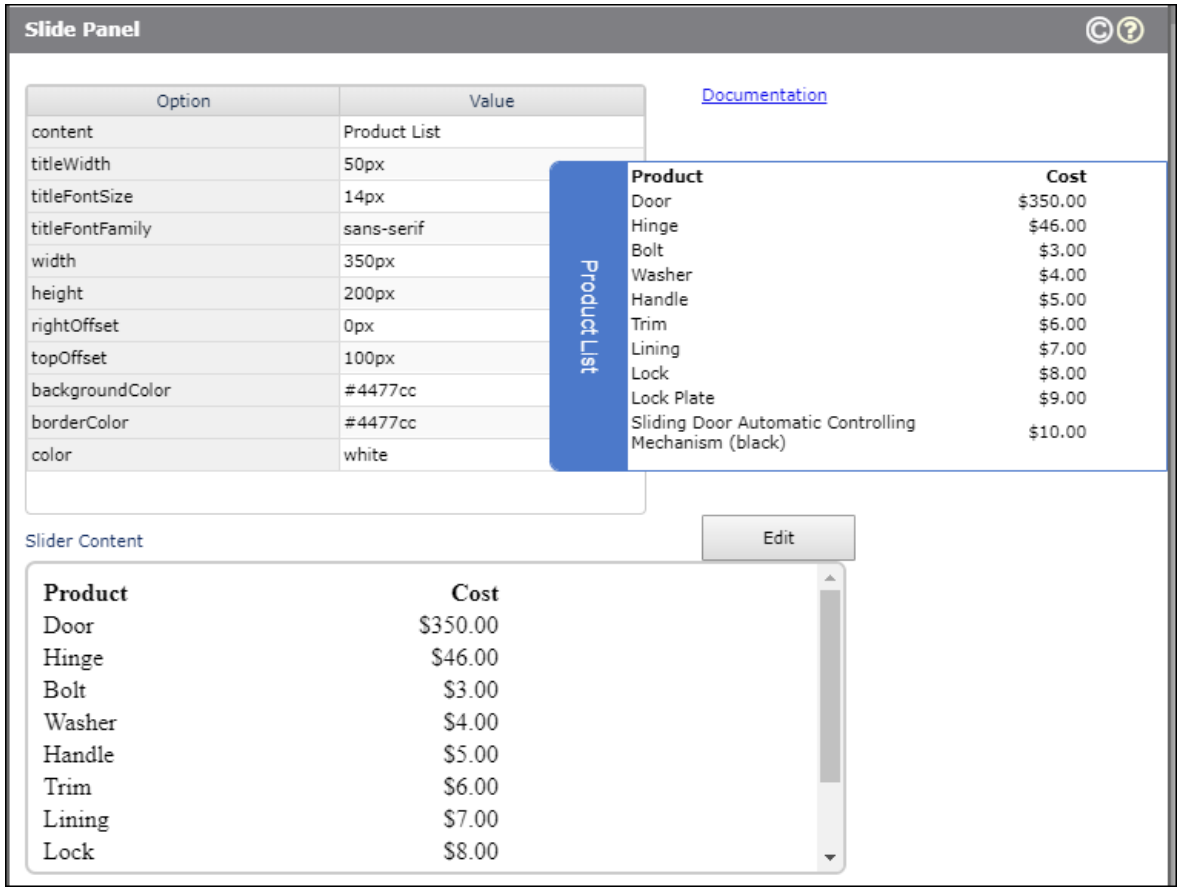
A Slider Panel can be included in both RD and non-RD DesignBais forms.

Refer to the DesignBais Responsive Design Manual section Implementing a Slider Panel if you want to add a slider panel to a Responsive Design form.

For a non-RD DesignBais form there is a Slider demo form DBDEMO*SLIDEPANEL, which is shown below, in the DBINET.DEMO account. The form and the basic code behind it are available by clicking in column 1 of the *Slide Panel* row of the Demo Forms list which can be displayed by selecting the Demo Forms menu option on the top menu of the Development Tools form.

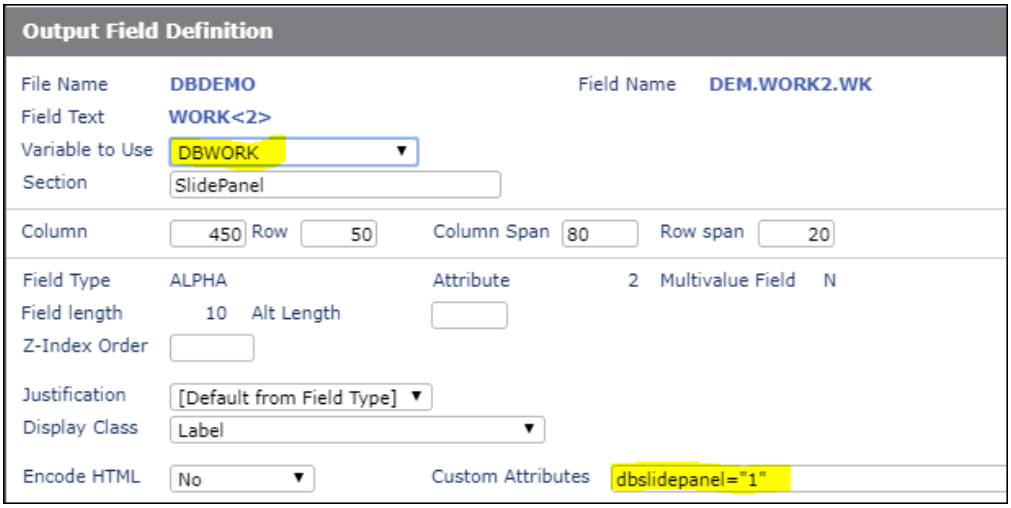
The basic code used to drive the display of the slider panel is contained in the DBLIB subroutine DBI.I.DEMO which is included in the DesignBais release in account DBINET.DEMO.

Non-RD DesignBais form

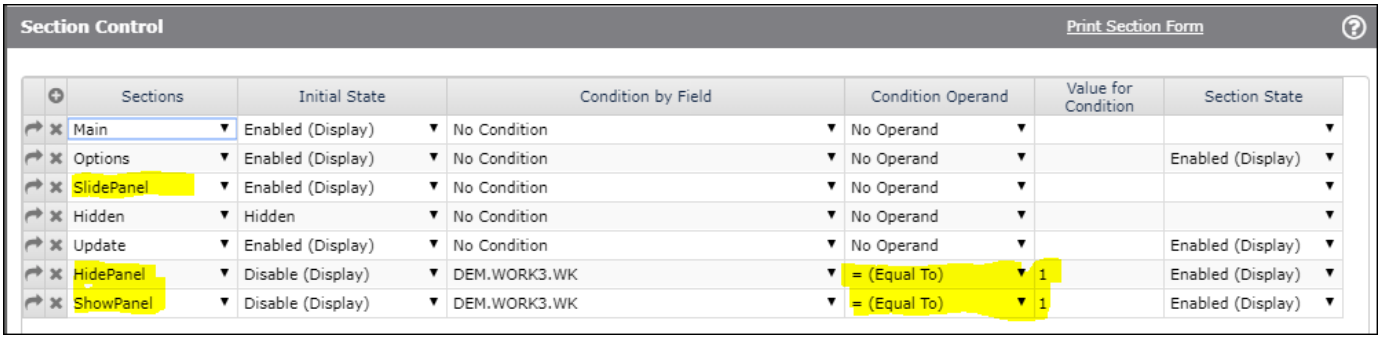


The snip above shows an example of a slide panel on a DesignBais demonstration form. This example includes the setup fields for the panel. In a real application these fields would not be visible to the user. The slide panel content is derived from a work field that is placed on the form. The Slide Panel functions in DBI.G.AJXCMD control the slide panel and will set the custom attribute value Custom Attributes = 'dbslidepanel="1"', if it is not already present, on the work field in the DesignBais form.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3



The content of the work field will be displayed on the slider panel. This can be text or HTML code.



In the AFTER DISPLAY event the slider panel can be set up. In the BUTTON event the slider panel can be updated, or revealed or hidden. Note that a slide panel cannot be built and displayed in a single call to the server. A possible approach is to build the panel in a validation event and set DBBUTTONCLICK. On the next button event the panel can be displayed using the “open” parameter (see below).

Refer to the DBLIB subroutine DBI.I.DEMO to see the basic code required to implement the slider panel.

The content of a slider panel is built by the developer.

If developers wish to construct HTML elements and assign them elements ids that the DesignBais engine can identify then it is best to adopt the OFR element naming convention:

tdxspan84v1x3z1 = “tdx”:“elementId”:“v”:DBWLEVEL:“x”:row:“z”:col

Clicking on the element in the on-form report will return the row and column position of the grid element.

See OUTPUT.TYPE regarding the requirements for a date input OFR cell.

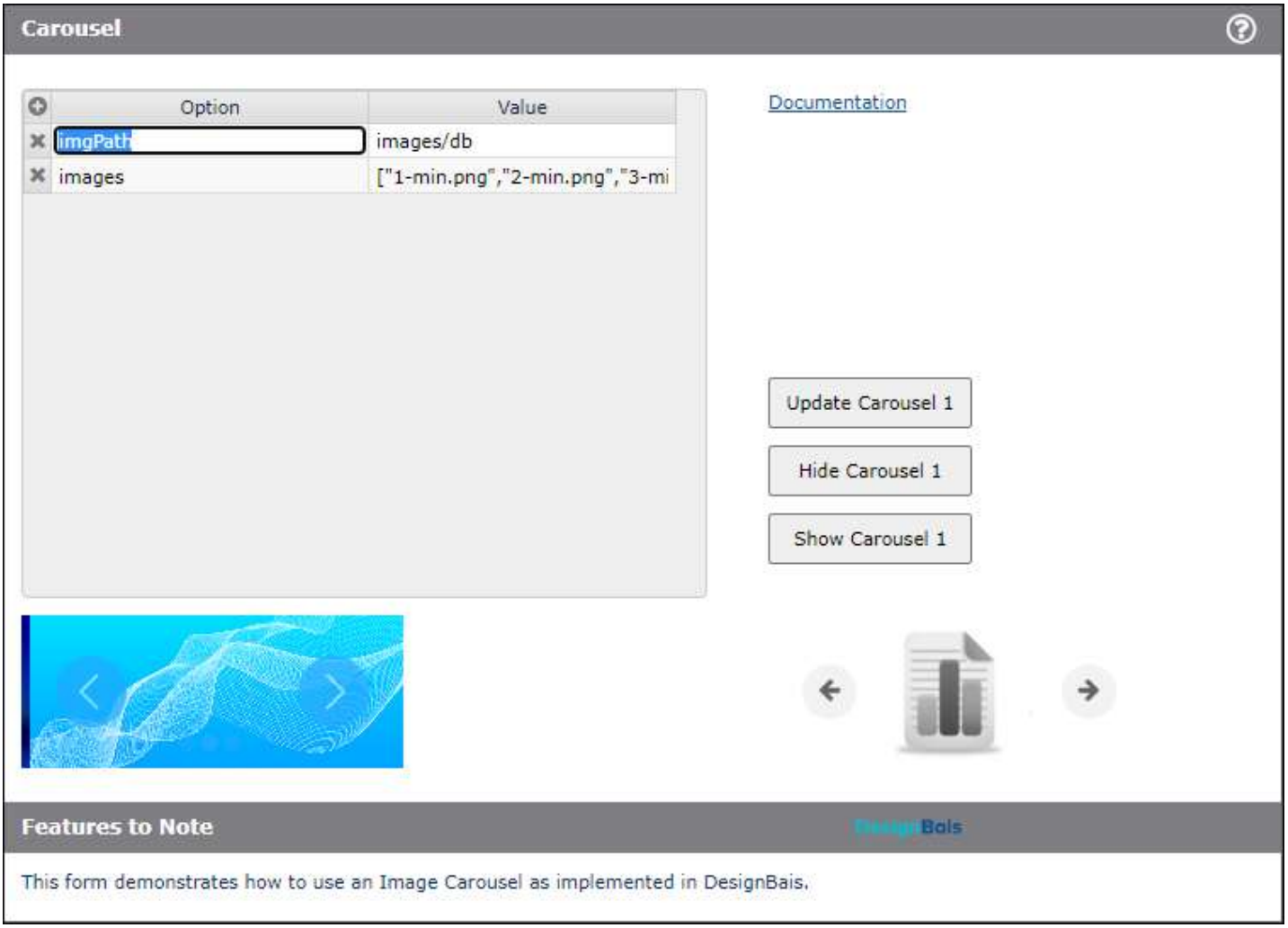
Implementing a Carousel

A Carousel can be included in both Responsive Design and Classic (non-RD) DesignBais forms.

Refer to the DesignBais Responsive Design Manual section Implementing a Carousel if you want to add a carousel to a Responsive Design form.

For a non-RD DesignBais form there is a Carousel demo form DBDEMO*CAROUSEL, which is shown below, in the DBINET.DEMO account. The form and the basic code behind it are available by clicking in column 1 of the *Carousel* row of the Demo Forms list which can be displayed by exercising the Demo Forms menu option on the top menu of the Development Tools form.

The basic code used to drive the display of the carousel is contained in the DBLIB subroutine DBI.I.DEMO which is included in the DesignBais release in account DBINET.DEMO.



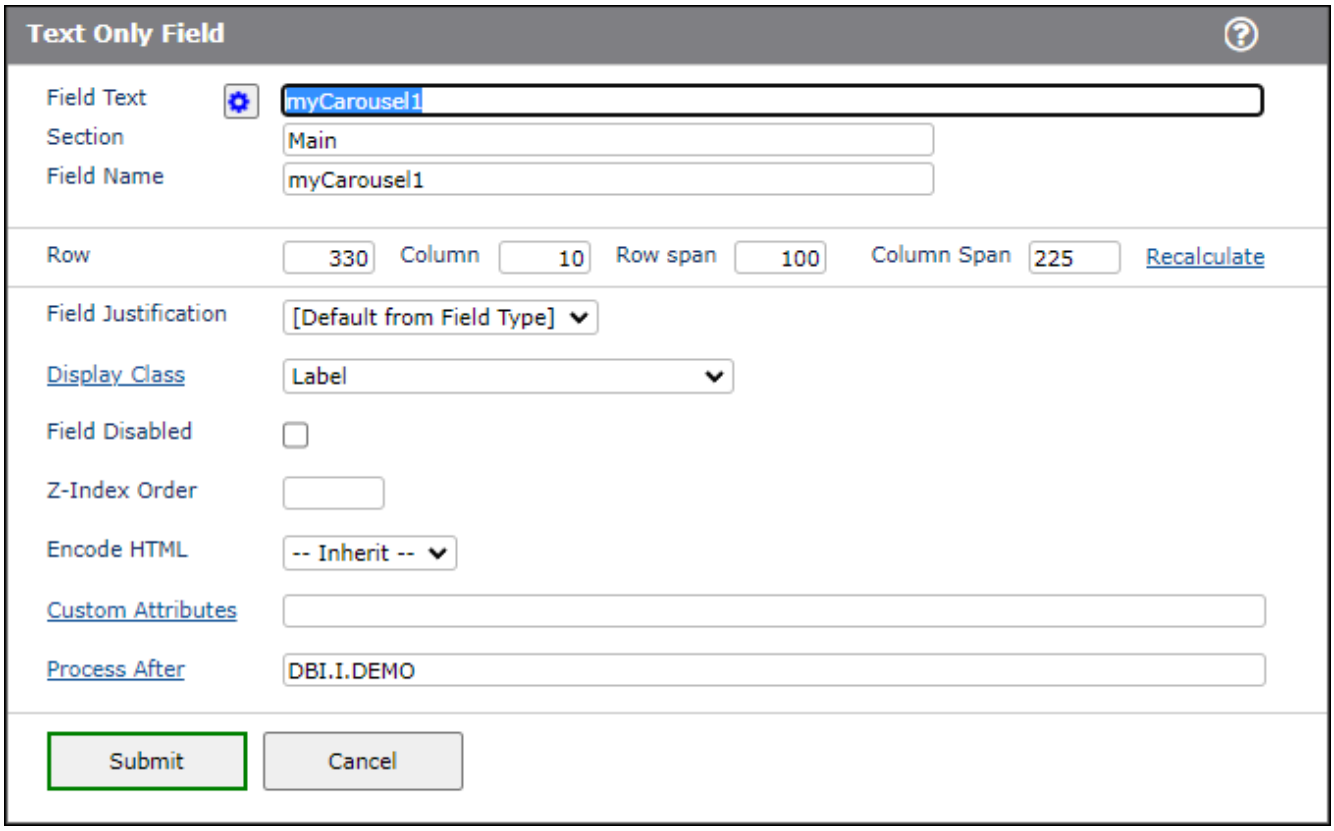
The carousel on the left (above) is an example of a carousel that has scroll buttons to allow the user to scroll through the images. The carousel on the right is clickable and demonstrates the ability to pass control to another form depending on which image is clicked. Refer to the basic code behind the buttons on the form.

To implement a carousel you will need to add a text only field to the form. The ID of the field is used by the javascript command to populate it with the carousel.

The Process After on this field will be invoked when a carousel image is clicked and the callback option is included in the carousel options. Refer to *callback* in the the full list of options to control the carousel display shown below.
The Process After EVENTSOURCE is the Id of text field. The carousel Image Name must be passed in PROCESS.PARAMETER.

Show and hide are implemented using the DBI.G.AJXCMD routine. For example to hide the carousel use:

```
AJX.DATAIN = 'myCarousel1'
AJX.DATAIN<2> = 'row'
CALL DBI.G.AJXCMD('HD',AJX.DATAIN)
```



Carousel Options

Option	Allowed Values	Description
imgPath	String	Relative path to the folder containing the carousel images. Default: "images"
images	String array	Comma delimited image names. Default: ["1-min.png", "2-min.png", "3-min.png"]
showButtons	Boolean	Show left/right navigation buttons Default: true

Option	Allowed Values	Description
showDots	Boolean	Show navigation dots Default: true
skin	String	Various preset skins. Possible values are: blue, red, green, orange, gray, dark and white. Default: "blue"
buttonType	String	"arrowhead" or "awesome". Default: "arrowhead". **Note: "awesome" can be used if FontAwesome is loaded on the page.**
buttonWidth	String	Width of navigation buttons in px. Default: "40px"
buttonColor	A color value (e.g. blue)	Fore color of navigation buttons. Default: #eff
buttonBackgroundColor	A color value (e.g. blue)	Background color of navigation buttons. Default: #2af
buttonBorderColor	A color value (e.g. blue)	Border color of navigation buttons. Default: transparent
buttonActiveOpacity	A decimal number (0 to 1)	Button opacity when not active Default:0.7
buttonOpacity	A decimal number (0 to 1)	Button opacity when active (mouse over) Default:1.0
buttonOffset	String	Offset of navigation buttons from the left and right edges. Negative values will make the buttons appear outside by making the images narrower. Note that images are not distorted but the aspect ratio is lost.
dotWidth	String	Width of dots. Default: "10px"
dotsTopOffset	String	Offset of navigation dots from the bottom edge. Default: "-20px" Negative values will make the dots appear outside by making the images shorter. Note that images are not distorted but the aspect ratio is lost.
dotColor	A color value (e.g. blue)	Color of naviagtion dots. Default: "#2af"

Option	Allowed Values	Description
dotActiveOpacity	A decimal number (0 to 1)	Dot active opacity (i.e. the corresponding image is being displayed) Default:1.0
dotOpacity	A decimal number (0 to 1)	Dot opacity when not active Default:0.5
animationSpeed	Number	Transition time from one image to the next in millisconds. Default: 700
loopPeriod	Number	Wait time between image transitions. Default:2000 **Set this number to 0 to stop animations.**
callback	String	Callback function name. The function is called on tap, click, swipe etc. events. Default: "" (i.e. no callback).
height	String	Carousel height in px Do not specify if responsive
width	String	Carousel width in px Do not specify if responsive
leftOffset	String	Offset of the carousel from the left of its relatively positioned container or document body. Default: "0px"
topOffset	String	Offset of the carousel from the top of its relatively positioned container or document body Default: "0px"
aspectRatio	Number	The width/height ratio of the carousel when the width is specified in pixels OR when the carousel is placed in a responsive grid column without specifying a width.

Clearing a Field using a Basic subroutine

Under certain conditions you may encounter a problem trying to clear a field using basic subroutine logic.

Consider the case where a new record is being created using a maintenance form. Most fields on the form will be initially empty (some fields may have default values).

As the user enters data into the form the browser holds these values but the database has no record of them since the data has not yet been passed back to the database via, say, a write of the record after the user clicks the submit button.

The form may have a button designed to clear values that have been entered by the user. This could be implemented to make the form more functional, by allowing the operator to quickly clear several fields by merely clicking a button. This button calls a subroutine with code similar to:

```
39 BUTTON.CLICK:
40    BEGIN CASE
41       CASE EVENTSOURCE = "B.CLEAR"
42          DBRECORD<4>=''
43          DBRECORD<5>=''
44    END CASE
45    RETURN
```

This works fine if the record is existing and has been read from the database. But it does not clear the fields if it is a new record.

There are three options to solve this:

1. Put a “Process After” on each field, even if the process does nothing but cause a server hit.

2. Have a “pre-clear” button which does a DBBUTTONCLICK on the real clear button. The event on the “pre-clear” will pass all of the data back to the database so that the “clear” will work.

3. Use the javascript functions in DBI.G.AJXCMD to do your own clear of the browser element.

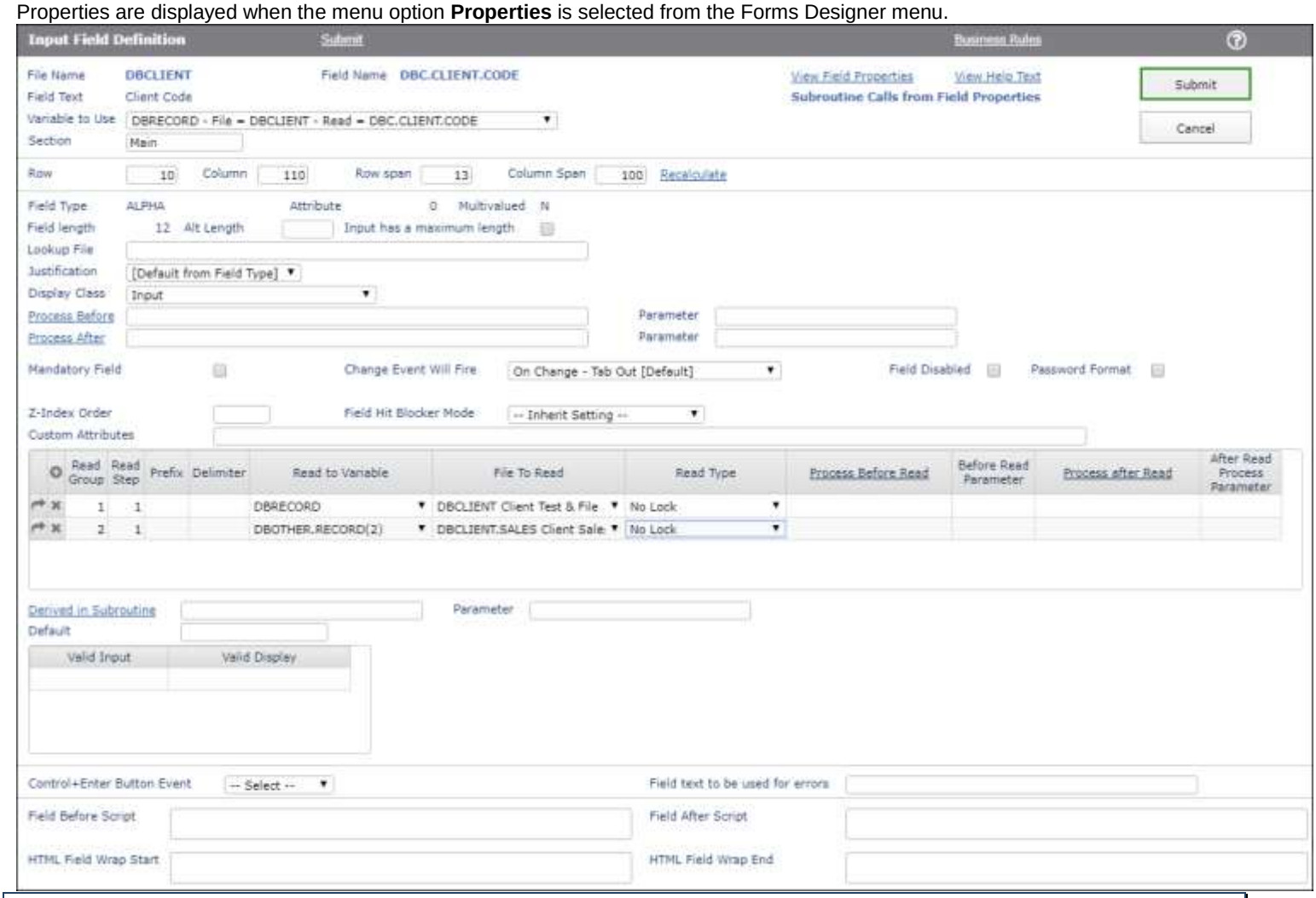
```
AJX.FUNC   = 'jqsv'            ;* jQuery Set Value
AJX.ARG    = 'DBC.CLIENT.NAME' ;* Field Name (DesignBais extracts the xml ID)
AJX.ARG<4> = ''                ;* Not needed but would hold the value to be pushed into the browser
CALL DBI.G.AJXCMD(AJX.FUNC,AJX.ARG)
DBRECORD<4> = ''
```

The data controlling the returned data is actually held in the SCREENREC common variable. For DesignBais to recognise a change it has to have been updated with the value entered before DBRECORD<4> is cleared. The update happens after the program logic.

Forms Designer Properties (Menu Option)

Various properties are associated with fields on a form. The following describes the available properties and what controls the properties pertain to.

Properties are displayed when the menu option **Properties** is selected from the Forms Designer menu.



Properties:

Position, width and depth Properties

Field Length — Input and Output Fields

Display-only fields loaded into the form from a Field Property (file DBIPROP).

Alt Length — Input Fields

Override for number of characters which may be entered into an input field. If the 'Input has a maximum length' field is checked and this field is not null, then the maximum input length is set to this value.

Input has a maximum length — Input Fields

If set, will force a maximum input length. If Alt Length is not null, that value is used as the maximum length, otherwise the 'Field Length' value is used.

Row — All controls

Controls the row (Pixel on the y-axis) that the control starts in.

Column — All controls

Controls the column (Pixel on the x-axis) that the control starts in.

Row Span — All controls except MultiValue fields.

Controls the depth (height) of a field in Pixels. To change the height of a MultiValue control, select the Multivalues option from the side menu.

Column Span — All controls except the Image control

Controls the width of the field in Pixels. If the field is a Text Only label control, the field will wrap if its contents do not fit in the width entered.

MV Input or Output lengths are calculated as follows:

- Start with Property Length
- Allow for Valid Input List code or Description lengths that may be longer
- Calculate width in pixels: length x font size / 2 (default font size = 8)
- If the calculated text heading width is larger than the field width then use text heading width

On text, output and report type fields the column span can be a designated as a percentage of the form width. If set to “100%” then the resulting HTML will then include “right:0px” which causes the field element to span from the starting column position to the right hand margin of the form. Note that use of “100%” still allows an element to commence a a column position greater then 0.

Recalculate — From Release 8.4.1.1 the Forms Designer has been enhanced to respond to the display classes within the style group that has been assigned to the form. The developer can now use the *Recalculate* button to recalculate the *Row Span* value based on the *Display Class.*

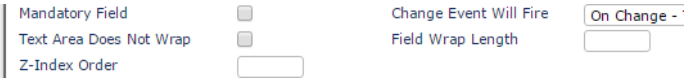
The developer can still amend the row span to any desired value. The *Recalculate* button references the style record for the display class and sets the row span based on the font or the height element (e.g. height: 12px).

Text Area Additional Fields

Text Area Does Not Wrap — For a field that is added to a form as a text area (Field Property ‘Multivalued’ is checked), there is an option to not have the text wrap at the end of the line. A new line will only occur when the end-user presses the Enter key.

Field Wrap Length — There may be instances where a value mark is to be entered exactly at the line wrap point. This can only be used where the field has a display style that utilizes a fixed pitch font Like Courier or Courier New.

The Field has to have a column span that exactly fits the number of characters in its width, otherwise the value mark will be placed in an incorrect position.



Display Properties

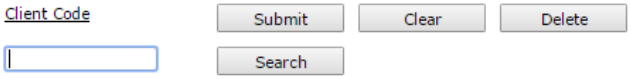
Display Class — **Input and Output Fields**

Sets the display characteristics for the field. The names used correspond with the Style Group settings – Input fields display a selection of input classes and Output fields display label-type classes. Note that from Version 7 onwards Display Class is case sensitive.

Display Class — **Buttons and images**

If manually specified, the style must be present in file DesignBais.css in the Web component on the web server. If left null the style defaults to the style specified in the Button Class in the Style Group assigned to

the form. If the Button Class in the Style Group is null, the Default Style is used called BUTTONdbaisDefault. The default class is shown below:

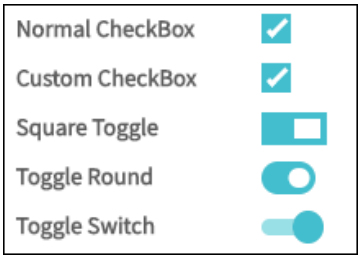


DesignBais comes standard with a Hyperlink style named: **dbaisSearchLabel** shown as Client Code above. Note that from Version 7 onwards Display Class/Style is case sensitive. All button display class names **must** commence with the uppercase string BUTTON.

From Release 8.4.1.1 there is a lookup available for display class. Click the Display Class link.

Display Class

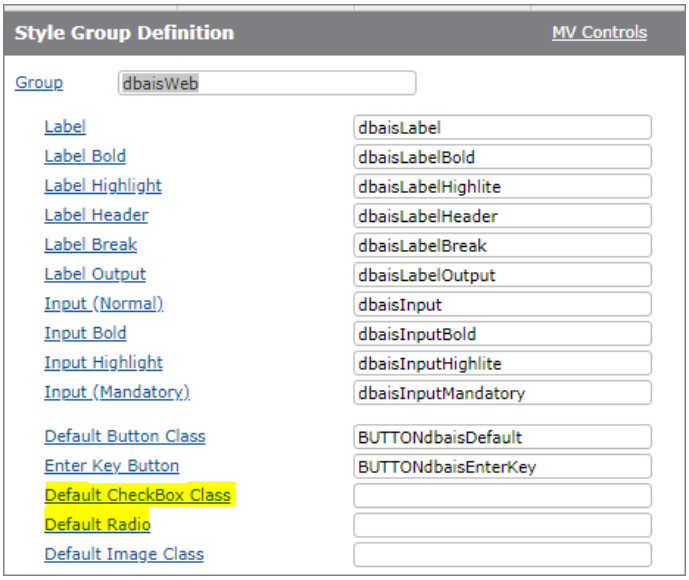
Check Box



From Version 8.3.3.6 onwards a display class can be applied to a check box in order to style the look. Some examples are shown to the right.

A standard check box is just an input element and this is what you get if your style group has no class defined in the *Default Check Box Class*.

If you wish to style your check box then you can add a *Default Check Box Class* to your style group. See right.



Alternatively you can specify a class on each check box on your form. In this case DesignBais expects the style definition to contain a label, an input, which is hidden, and a span which gives the desired look to the check box. See below.

```
▼<label id="lblcheckbox28v1" class="I2Toggle" style="height: 20px; z-index: 1; position: absolute; left:
403px; top: 590px; width: 40px; display: block; cursor: pointer;"> == $0
    <input tabindex="5" type="checkbox" style="display: none; top: 590px;" onclick="vs(event);" id=
    "checkbox28v1" value>
  ▼<span class="I2Togglespan">
      ::before
    </span>
  </label>
```

DesignBais releases several check box display classes eg. dbaisToggle, dbaisToggleRound, and dbaisToggleSwitch.

If, in Forms Designer, a display class is used for a check box that does not have the required structure as detailed above then the check box element will disappear from the grid. In this case use your mouse and hover over the row and column position of the element. The mouse hover title will display and you can then use Ctrl + Enter to display the field definition.

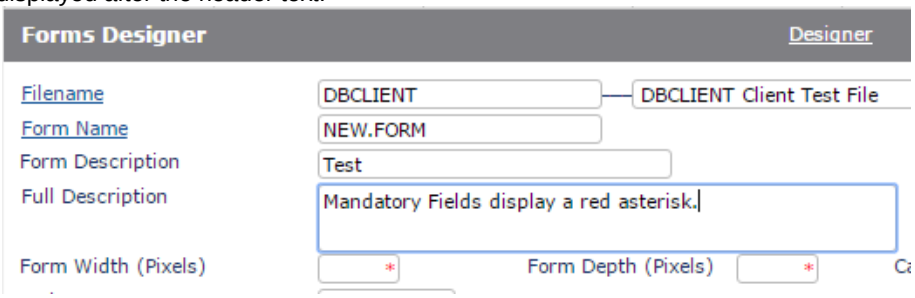
Password Format

Input Fields. When checked this will echo entered characters with an '*'.

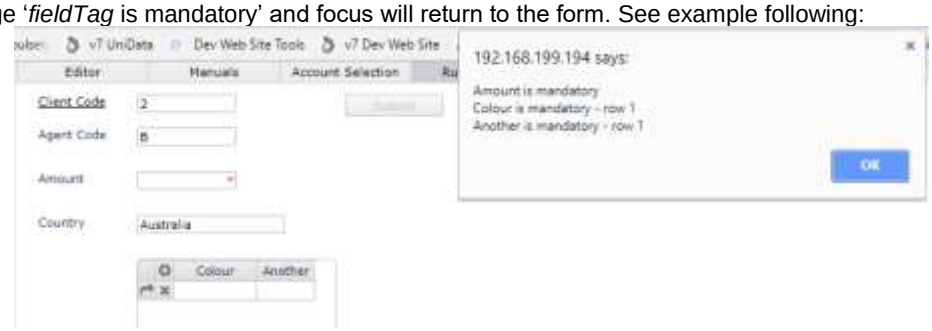
Mandatory Field

Input Fields and MV Grid Fields

When checked this field requires data. When mandatory fields are present on a form there will be a red asterisk displayed in the field until a value is entered in the field. Mandatory MV Grid fields will have a red asterisk displayed after the header text.



When a field is flagged as mandatory, and the field has no value (is null), and DesignBais is controlling the update of files via the submit button then when the submit button is clicked the browser will display a message '*fieldTag* is mandatory' and focus will return to the form. See example following:



When entering data on a form a similar message will display when a mandatory field is set to null. Focus will return to the field. The operator may then tab through the field leaving it null. The mandatory message will display on submit.
If DesignBais does not control the update then mandatory checks are not done on submit. The developer must include checks for mandatory fields in the basic code that handles the file update.

If you wish to remove the display of the red asterisk then:

To remove the yellow marked ones add a Style Definition *dbrequried* as described below. This will add to the *DesignBais.css*:

```
.dbrequired {
    display:none;
}
```

To remove the black marked ones change your *.dbaisMVHeader sup* as follows :

```
.dbaisMVHeader sup {
    color: red;
    vertical-align: top;
    display:none;
}
```

For date type fields the DesignBais data component is setting an inline style: "display:block". In this case you will need to add *!important*.

i.e. *display:none !important;* for both classes.

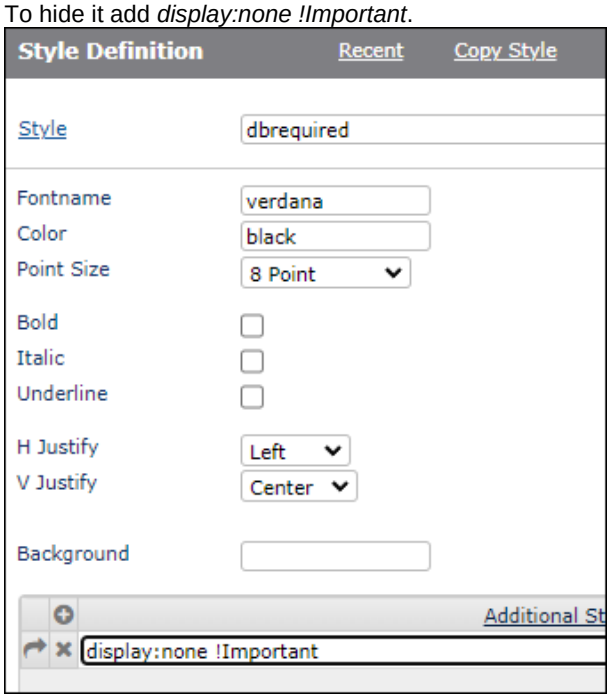
By default, it is the web component that adds the red asterisk to a mandatory field with a class-"dbrequired" as per:

*

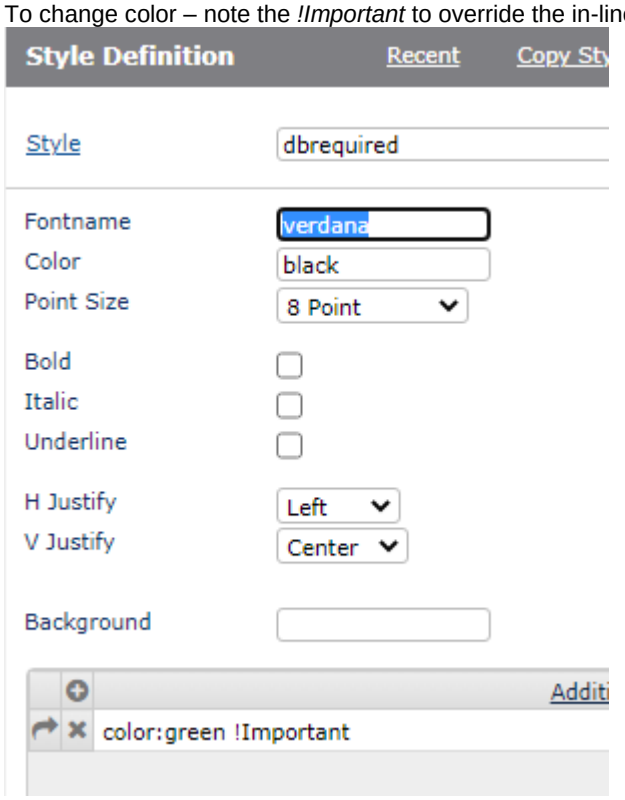
This is done from javascript and hits all dbMandatory fields.

There is no dbrequired style and hence to change the asterisk you can create a style in DesignBais.

To hide it add *display:none !Important*.

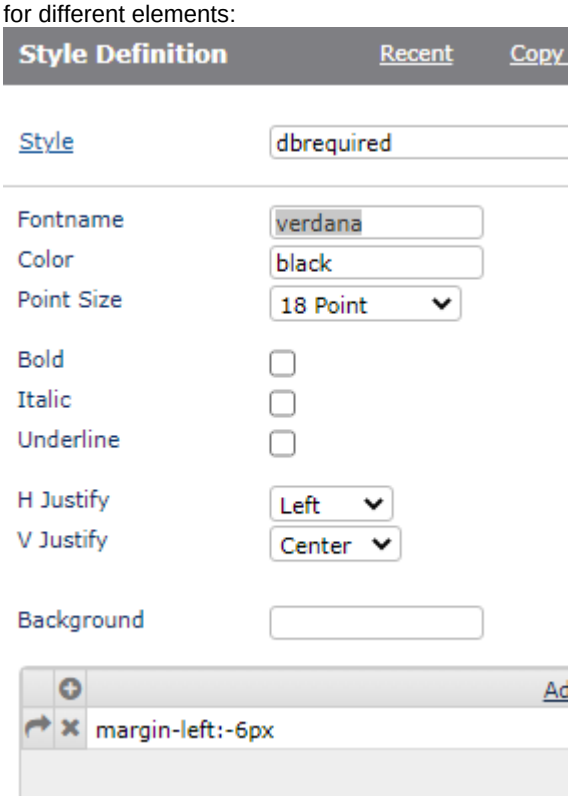


To change color – note the *!Important* to override the in-line style applied by the javascript:



The red border comes from the Style Group “Input (Mandatory)” style and could be changed in that style.

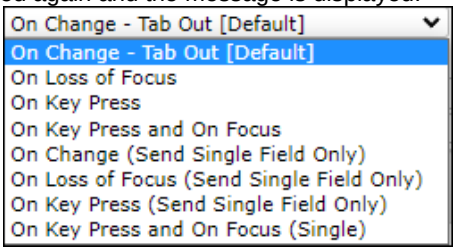
The asterisk can be made larger but this requires moving it with a margin setting – this may get complex for different elements:



The element has an id like “*text8v1div*” and could be modified using javascript but timing will be an issue to fit in with the setting provided by the web component.

Change Event Will Fire

The available settings for this field are shown below. The normal setting used in nearly all cases is the default ‘On Change – Tab Out’. Other settings can be used where appropriate. The setting ‘On Loss of Focus’ is referred to as an ‘On blur’ event and should be used with caution. If a validation uses on blur and an error message is displayed, and focus is passed back to the same field, then a loop is established which prevents the user from exiting this field. The on blur event displays a message, the user clicks OK to the message, and focus passes back to the field. Then, as soon as focus leaves the field, the on blur event is triggered again and the message is displayed.



Field Disabled

All fields

This field determines the initial state of a field at load time. You can enable a field at run time by setting DBENABLEFIELD in your Basic subroutine. DBENABLEFIELD may be multivalued to enable and/or disable more than one field at one time. Refer to “Controlling Sections at Run Time” in this manual. Note that fields flagged as multivalued, even if not in a grid, such as a text area field, will not appear as disabled. The field is disabled but can gain focus in order to allow the scroll bar, if present, to be activated. If a text area field is to appear disabled like other non-multivalue fields, then you will need to copy the field property to another name and remove the multivalue flag.

Z-Index Order

All Fields
By default, DesignBais layers images based on the X and Y position. The higher X and Y values the more towards the foreground the image appears. If a numeric value is entered in this field, this will override the default behaviour and place the image more towards the foreground. The higher the value, the further forward the image is displayed.

Field Hit Blocker Mode

All Fields
"Hit Blocker Mode" is a new feature in DesignBais that determines the browser's ajax request modes. The major modes are:

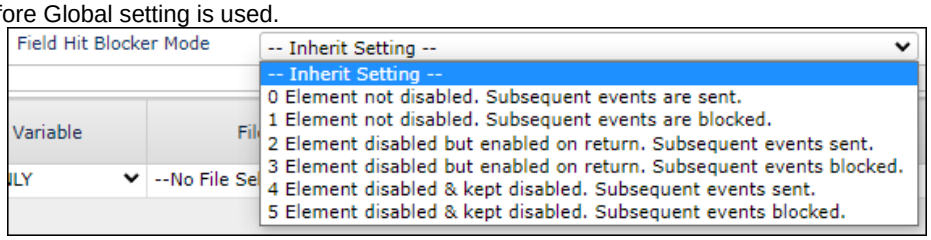
- No Blocking: This emulates DesignBais pre V7 IE-only mode. Events are queued and sent in correct order.

- Block Subsequent Events: This emulates DesignBais pre V7 cross-bowser mode. All subsequent events are cancelled if there is already an event that's being processed. Events are not queued. This mode is useful in situations where the state of a form depends on the result of the request that's being processed.

Together with these, it is also possible to disable the element (e.g. a button) that is raising the event and keep it disabled after the event has been processed.

In migrating from v6 (for both IE-only & cross-browser); we recommend the "No Blocking" mode as the initial setting. "Block Subsequent Events" can be used on individual elements if ncessary.

Field Hit Blocker Mode may be set Globally or on System Parameters. Applied at Field Level, then System Level before Global setting is used.



Values are:
" = Inherit Setting
0 = Element not disabled. Subsequent events are queued and sent in correct order.
1 = Element not disabled. Subsequent events are blocked. Events are lost and page is disabled.
2 = Element disabled but enabled on return. Subsequent events queued and sent in correct order.
3 = Element disabled but enabled on return. Subsequent events are blocked.
4 = Element disabled & kept disabled. Subsequent events queued and sent in correct order.
5 = Element disabled and kept disabled on return. Subsequent events are blocked.

User input is allowed for settings flagged as “Subsequent events are sent” but developers should note that in asynchronous mode user input may continue during the server hit but may be overwritten by data returned from the server hit. If individual field events affect the page flow data entry errors may occur. To avoid these issues perhaps all data should be validated when the page is submitted.

Assume an input field with a change event and a button with a click event on a form. The user makes a change in the input field and clicks the button. Here is what happens for each of the attributes listed above:

ID	Change Event fired	Click Event fired	Text hidden until return	Hide Text on return
0	yes	yes	no	no
1	yes	no	no	no
2	yes	yes	yes	no

3	yes	no	yes	no
4	yes	yes	yes	yes
5	yes	no	no	yes

It's easier to understand this in binary mode:

111 (least significant bit is on the right)
ABC

A : block subsequent hits if true
B : Disable element on event if true
C : Keep the element disabled on return if true

e.g.

101 = 5
block subsequent hits
don't disable the element on event
disable the element on return

Custom Attributes

Input and Multivalue Input Fields, Output Fields, Buttons, Check Boxes, Radio Buttons

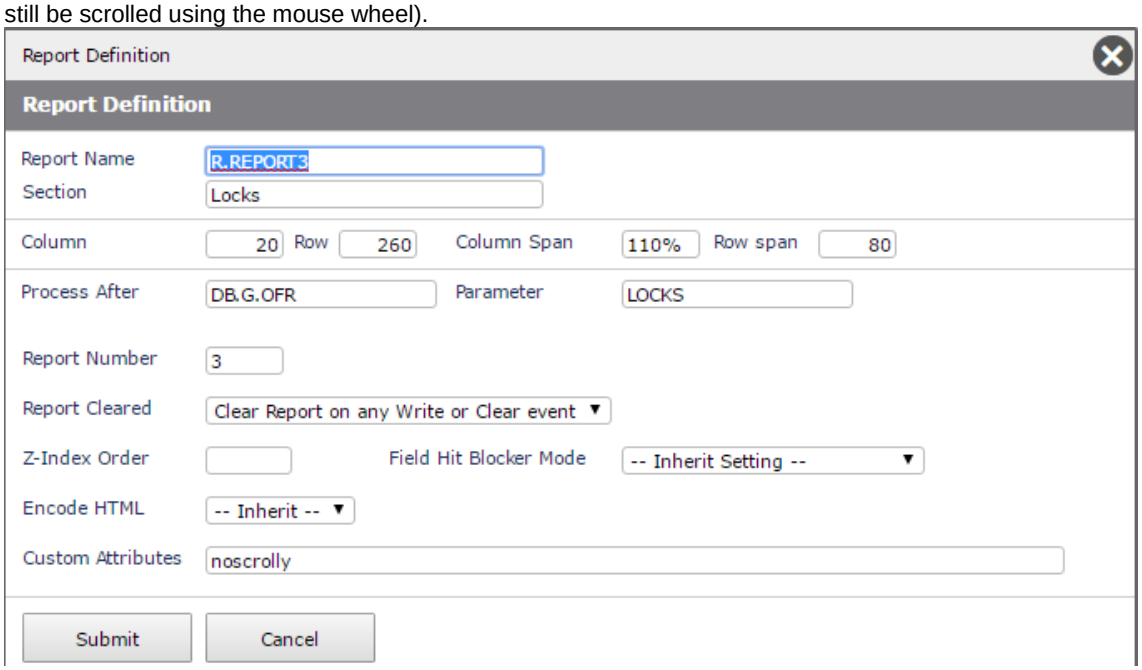
Provides the ability to include additional attributes in the HTML code generated by DesignBais. The developer can then act on this attribute from customised javascript. Note that multiple custom attributes can be assigned to a field, separated by a space.

Important: Custom attributes are case sensitive. The developer must apply the value with case as documented in this reference manual.

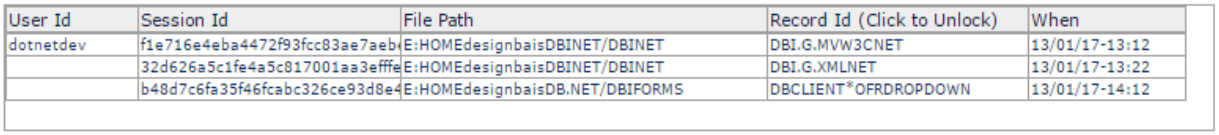
The following is a list of Custom Attributes used by DesignBais. These can be accessed in the *Custom Attributes* menu option in System Parameters.

Custom Attribute Description	Custom Attribute
Limit width of dropdown list	dbsellimit="1"
Onform Report remove horizontal scrollbar	noscrollx
Onform Report remove vertical scrollbar	noscrolly
Onform Report remove both scrollbars	noscroll
Tab into MV grid field with click event on process after/textarea output only field	readonly="readonly"
Invoke on-form html editor from input or textarea field	onformeditor="0"
Allow multiple selections from input field dropdown list	multiple="multiple" size="n"
Convert dropdown selection list to scrollable list with n elements	size="n"
Output field housing a slide panel	dbslidepanel="1"
Template for input field placeholder text	placeholder="Username" autocomplete="off" autocorrect="off" autoca
Skip field during autofocus	dbnofocus
Context menu	oncontextmenu="getMouse(event);event.preventDefault();"
Field Place Holder	placeholder="place holder text"
Field Title activated by mouseover	title="field title text"
Google Address field	geoloc="true" address=""
Google Address field with country filter	geoloc="true" address="" countryFilter="AU|NZ"
Suppress the date calendar from automatically popping up	showdbon="none"
Generate a blur or change event on tab out even when field is null	fireonblank="true"
Adjust Output Field Height based on the data (when text wraps)	dbexpandable
Select options will have the same class as the select and will not be rebuilt after initial display	dbsetoptionclass
Cancel tab-out event and fire the "change" event on the form element	onkeydown="if (event.keyCode==9){event.preventDefault();event.ato
Set field minimum value	min=""
Set field maximum value	max=""

Example showing how to suppress the vertical scrollbar on an On-form Report. (Note that the report can still be scrolled using the mouse wheel).

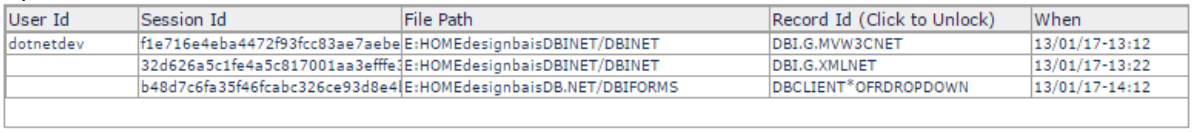


The report with no custom attribute entry shows, on the right hand end, a white space available for the scrollbar.



User Id	Session Id	File Path	Record Id (Click to Unlock)	When
dotnetdev	f1e716e4eba4472f93fcc83ae7aeb	E:HOMEdesignbaisDBINET/DBINET	DBI.G.MVW3CNET	13/01/17-13:12
	32d626a5c1fe4a5c817001aa3efffe	E:HOMEdesignbaisDBINET/DBINET	DBI.G.XMLNET	13/01/17-13:22
	b48d7c6fa35f46fcabc326ce93d8e4	E:HOMEdesignbaisDB.NET/DBIFORMS	DBCLIENT*OFRDROPDOWN	13/01/17-14:12

The effect of custom attribute ‘noscrolly’ is to remove the 18 px wide space at the right hand end of the report.

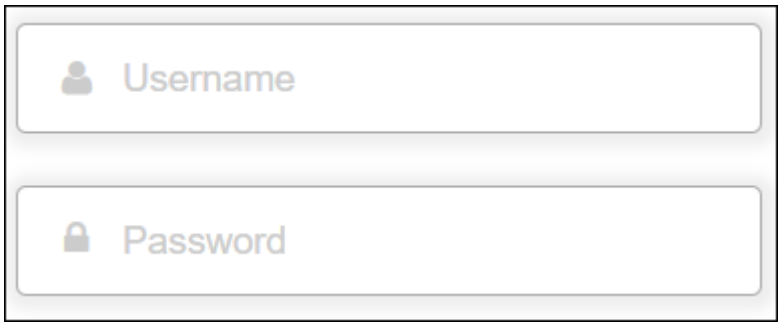


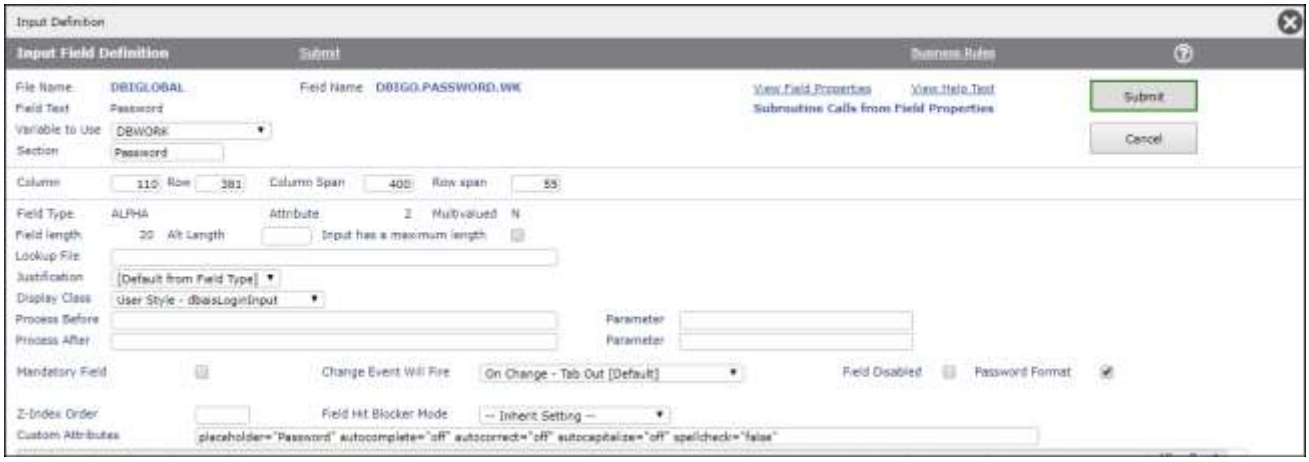
User Id	Session Id	File Path	Record Id (Click to Unlock)	When
dotnetdev	f1e716e4eba4472f93fcc83ae7aebe	E:HOMEdesignbaisDBINET/DBINET	DBI.G.MVW3CNET	13/01/17-13:12
	32d626a5c1fe4a5c817001aa3efffe	E:HOMEdesignbaisDBINET/DBINET	DBI.G.XMLNET	13/01/17-13:22
	b48d7c6fa35f46fcabc326ce93d8e4	E:HOMEdesignbaisDB.NET/DBIFORMS	DBCLIENT*OFRDROPDOWN	13/01/17-14:12

Example of using Custom Attributes to provide placeholder text in an input field. The entry in custom attribute is:

placeholder="Username" autocomplete="off" autocorrect="off" autocapitalize="off" spellcheck="false"

placeholder="Password" autocomplete="off" autocorrect="off" autocapitalize="off" spellcheck="false"





See Custom Attributes section.

Section Property

Section

All Controls

Determines the section that the control belongs to on a form. The default is **Main**. A section can be enabled, disabled, hidden or collapsed based on run-time properties associated with the controls on the form. The Section name is by convention entered as text case but this is not mandatory. The underscore character within a section name is used to designate a Subsection name and thus the underscore character should not be used in a section name other than to form a Subsection name. Subsections are only required when the parent section is flagged to collapse (see Collapsing Sections). There is no prohibition on setting up a Subsection for a parent that is not flagged to collapse but the developer may find that text type fields within the Subsection do not behave as required. Text type fields within a section that is disabled cannot be disabled, however they may be styled to indicate they are part of a disabled section (see Styling Disabled Text Fields).

Please see Sections later in this chapter.

Dropdown Lists and Defaults

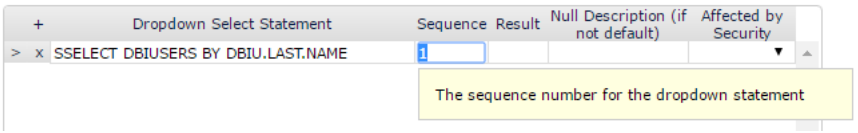
Lookup File

Input and Output Controls

Specifies a file name to supply a list of keys and descriptions for the field. The descriptions are rendered in a dropdown listbox. When the field being defined has a Lookup File entered in Field Properties, that data is used as a default for this value. Such a default may be overridden. When a file name is used here, DesignBais will build the list as per the definition in the File Properties form.

For file-based lookups use a Filename by itself, or use Filename,*sequence number* . Refer to File Properties. This example is for the DBIUSERS file and in this case use DBIUSERS,1

Dropdown Select Statements



There is also the ability to add variable-based lookups. To do this use a 'V:' prefix followed by a valid Field Property from the form file assigned to the DBWORK variable.

Eg. V:COUNTRY.LIST.WK

From a BASIC subroutine the format of the DBWORK variable must be:

DBWORK<Attribute,*List Counter,*1> = ID
DBWORK<Attribute,*List Counter,*2> = Description

Note that the ID for a date field dropdown must be the date in external format. A date field is expecting external date format as the value returned from the web – DesignBais can then apply the input conversion.

When a numeric field is used to form a dropdown list it cannot have a value in the *No of Decimal Places* field since for numeric fields with a value in the decimals field DesignBais appends a comma [,] to the conversion code. Setting *No of Decimal Places* to zero means no decimal points but sets the conversion as “MD0,”.

The effect of a conversion code containing the comma, for dropdown lists containing values above 999, is to have values such as “1,000” or “1,673”. The browser receives the following:

```
$("#select24v1").val(["1,673"]);
```

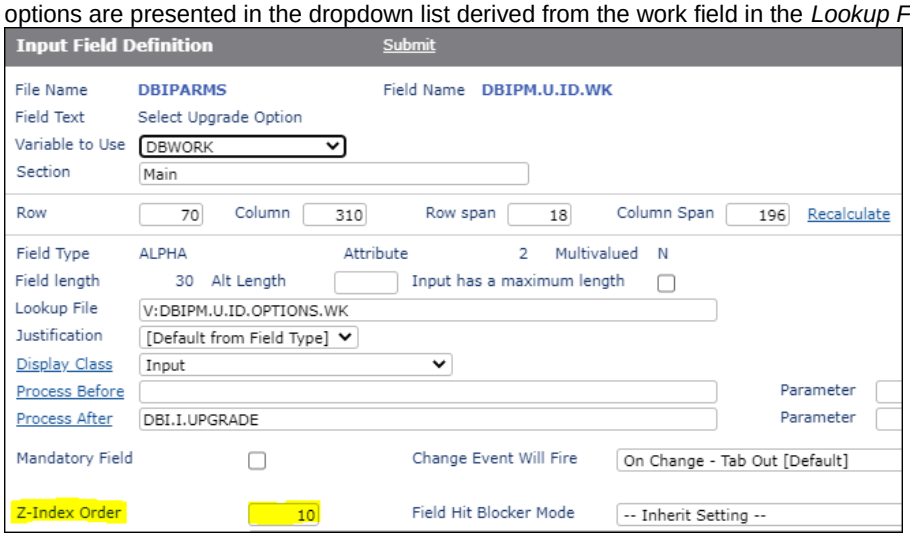
and does not display anything because 1,673 is not a valid value.

Typically the first value of, for example, a lookup for Country, would be:

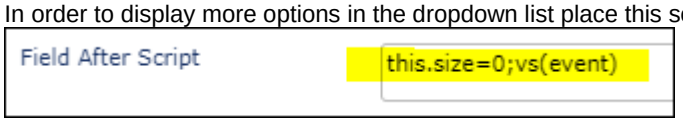
DBWORK<COUNTRY.LIST.WK,1,1> = “”
DBWORK<COUNTRY.LIST.WK,1,2> = “ – Select Country – “

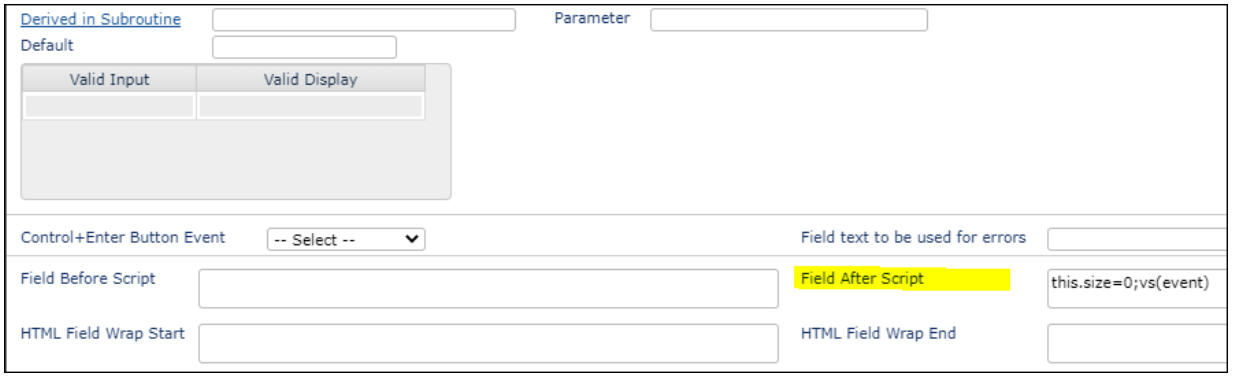
Note that you can use the Custom Attribute dbsellimit=”1” (refer to Field Properties) to limit the width of the field that the browser renders. The width will then be set to the field length as defined in the Field Properties. HTML control characters such as ‘&’ must not be used in the ID or Description.

The length of the dropdown presented by the browser can be controlled by use of the *Field After Script*. Consider the input field shown below, taken from the DesignBais Upgrade Logs form. By default about 19 options are presented in the dropdown list derived from the work field in the *Lookup File* field.



In order to display more options in the dropdown list place this script in the *FieldAfter Script*.





In the AFTER DISPLAY event for the form use the Ajax function to add an event as shown:

```
CASE SCREEN.NO = "U10"
   GOSUB CREATE.DROPDOWN
   * Set Dropdown Size when clicked
   AJX.FUNC = "addEvent"
   AJX.ARG    = "DBIPM.U.ID.WK"
   AJX.ARG<2> = "onmousedown"
   AJX.ARG<3> = 'this.size=this.length;'
   *
   AJX.ARG<1,-1> = "DBIPM.U.ID.WK"
   AJX.ARG<2,-1> = "onblur"
   AJX.ARG<3,-1> = 'this.size=0;'
   CALL DBI.G.AJXCMD(AJX.FUNC,AJX.ARG)
```

Notes:

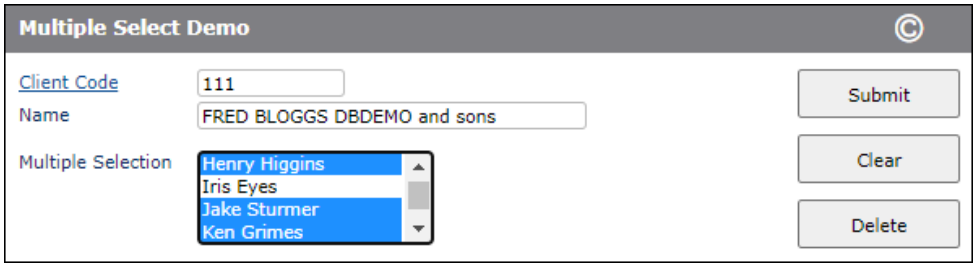
- Just setting size in a dropdown leaves it permanently open.
- The above example sets the size to “this.length” which is the number of options. A specific number can be specified.
- When the dopdown is clicked the “onmousedown” sets the size.
- Tabbing or clicking out of the dropdown invokes the “oblur” event which removes the size making the full list disappear.
- The user can still tab to the field and type the start of the dropdown value to select one from the list.
- DesignBais will add ‘v’:DBLEVEL to the field id if it is not present as required by the DesignBais engine.
- You may need to increase the Z-Index order value so that the dropdown displays above the form.

When a lookup file is specified for a field the dropdown list is built at form load time. If the form then allows new records to be created that are to be included in the dropdown list, these will not appear until the form is reloaded. To overcome this issue use the DesignBais DBDROPDOWN variable. Refer to the DesignBais Responsive Design Manual for details of how to code for this. The details are in the *Lookup* section.

Multiple selection from a dropdown list can be enabled by use of a custom attribute in the input field.
For example the custom attribute can be set to:

multiple=”multiple” size=”4”

The “4” indicates the number of options to display within the dropdown list.



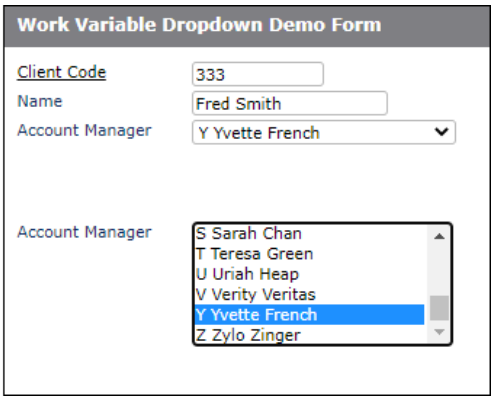
Multiple selections are made by pressing the Ctrl key and clicking dropdown options in turn, or by clicking in the dropdown list at one place, then holding the Shift key and clicking somewhere else in the list. In the latter case all options spanned by the two click positions are selected.

The selections are saved as multi-subvalues within multivalues.

Setting a custom attribute “size=n” in a field with a dropdown list limits the number of visible options to "n" but the element is no longer a drop-down, it's a scrollable list.

The first account Manager field in the image to the right is a normal dropdown list.

The second Account Manager field includes the custom attribute “size=”6”. This displays as a scrollable list displaying 6 elements at a tiem.



Default

Input and Output Controls

Controls the default input for a field. The field can either be a literal or a string with a prefix of ‘V:’

Valid ‘V:’ variable prefixes.

@DATE	Default to the system date.
@TIME	Default to the system time.
@WEBLOGON	Default to the current user identifier.
@DBIACCOUNT	Default to the current login account or directory.
DBSTORE(*N*)<*Attr,Val*>	Default to a value stored in DBSTORE.
DBRECORD<*Attr,Val*>	Default to a value stored in DBRECORD.
DBWORK<*Attr,Val*>	Default to a value stored in DBWORK.

Control+Enter Button Event

DesignBais provides for a hot key of Control+Enter to invoke a button function. Enter the name of the button to click when the Control+Enter keys are pressed. This can be useful to Invoke Search processes, calendars without having to resort to the mouse. If there is data in the field that the Control+Enter key is pressed, the change event (VALIDATE) will also be invoked.

This field is a dropdown list that contains names of all buttons on a form.

When a button name is selected, the user interface will enable the Control+Enter keystroke combination for that input field. When the user performs this key stroke combination, the button nominated will be clicked.

If this button loads a search form, the form will be displayed.

There is a routine provided from Release 4.3.3 onwards that will automatically set this feature up for your existing forms provided certain criteria apply.

Please make a back-up of your DBIFORMS file before running this procedure.

DBI.P.UPDATE.FORMS.CONTROL.KEY

This program (run from TCL), will process each form and set-up the Control+Enter button event for each button that has a Return to Field nominated. The actual return to field will then be flagged with the Control+Enter event.

This will not set-up the Control+Enter event for any button that has been nominated as a Submit, Delete or Clear button regardless of whether they have a return to field nominated.

Field text to be used for errors

Input and Multivalue Input Fields

Text from this field will be displayed in the case of a field error or warning. This is useful where, for example, a field is used in more than one form. The standard field display for that field may not be descriptive enough to use for browser messages. An example is the description applied to mandatory field warnings.

Field Before Script — Input and Multivalue Input Fields, Buttons

Javascript from the Field Before Script for a field on a form will be executed as the that field gains focus. See example below.

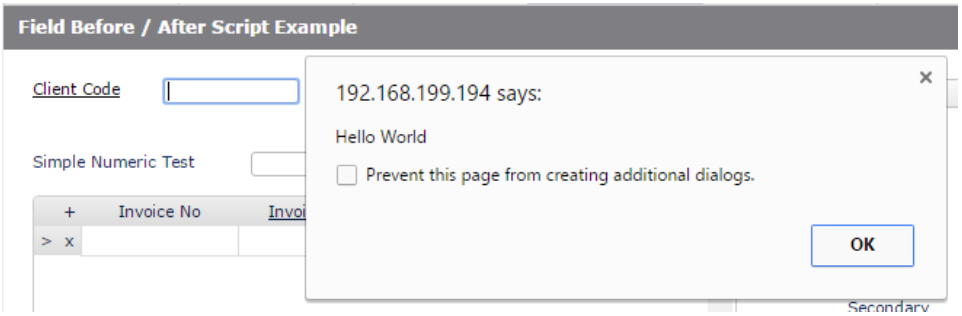
Field After Script — Input and Multivalue Input Fields, Buttons

Javascript from the Field After Script for a field on a form will be executed as the that field loses focus.

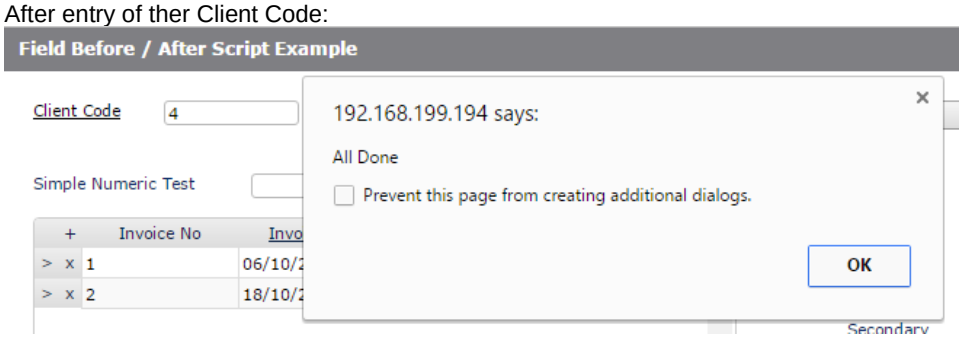
Example:

Script is added to the Client Code.

When the form is run the Before Script executes as the Client Code takes focus and the After Script executes with the standard after event, either on change or on loss of focus for example.



After entry of ther Client Code:



HTML Field Wrap Start — Input Fields and Buttons

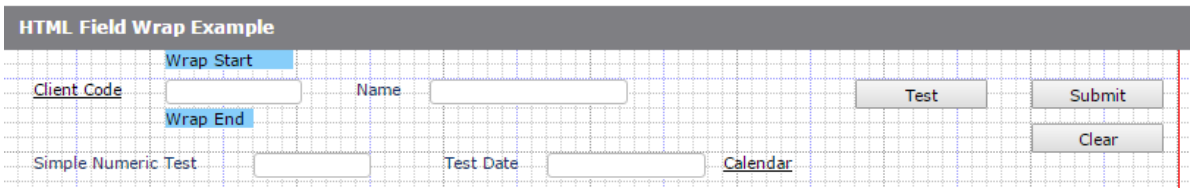
HTML code entered in this field will be slotted into the form layer “div” ahead of the DesignBais HTML code generated by DesignBais. See example below.

HTML Field Wrap End — Input Fields and Buttons

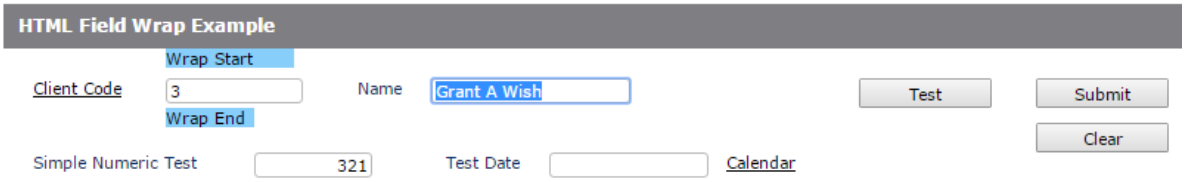
HTML code entered in this field will be slotted into the form layer “div” after of the DesignBais HTML code generated by DesignBais. See example below.

Example: The HTML code below is added to the Client Code input field in Properties within Forms Designer.

In Designer this is the result:



When the form is run this is the result:



Setting the Variable to Use

Variable to Use — Input and Output Controls

Used to determine which internal DesignBais variable is used for storage and retrieval of the field's contents.

One of the following valid values must be selected from the provided dropdown list:

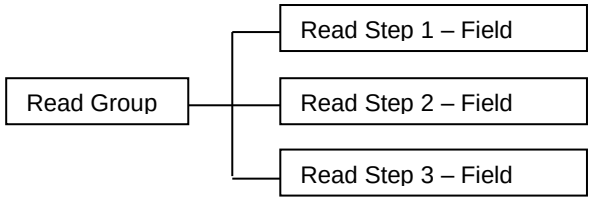
DBWORK, DBRECORD, DBOTHER.RECORD(1) to DBOTHER.RECORD(99)

Work Variables defined in Field Properties are referenced through DBWORK, regardless of the file in which the field is defined. Once a Work Variable is populated, the content may be used to read other records – see Read Group, Read to Variable, and related fields.

Controlling Reads

Read Group — Input Fields, Output Fields and MultiValue Fields

The Read Group is a numeric ID used to indicate that when a value in a field(s) changes, that a read is required. DesignBais uses a read group so that multiple fields (Read Steps) can be grouped together to form a key (delimited by Delimiters) allowing for multi-part key reads.
Eg.



When all of the Read Steps are filled, via a user input or another file read, a read takes place. Once all read steps are complete, any changes to any field in a read step will invoke another read.
The Read Group ID is chosen by the developer and must only be assigned to a single file for any given form or group of sub-forms. Data inconsistencies may occur if the same Read Group number is used for more than one file. Generally, Read Group 1 is used for DBRECORD, Read Group 2 is used for DBOTHER.RECORD(2), etc. This numeric agreement is not required but is useful.

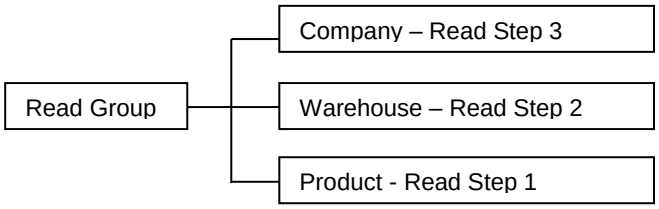
Read Step — Input Fields, Output Fields and MultiValue Fields

A Read Step is used to define the position of the field in a multi-part key separated by Delimiters. Read steps allow the developer to define the field position in the key, thus allowing keys to be placed in a different order on a form than they appear in a key.

Eg.

Key = Product * Warehouse * Company

On the form they are in the following order.



Keys which are not delimited use Read Step 1.
See **Key.Part** for information about coding related to Read Steps.

Prefix

Input Fields, Output Fields and MultiValue Fields

In some instances multi-part reads may have a pre-determined prefix.
To simplify reads with a multi-part key, a prefix may be entered. If the prefix has a delimiter, the delimiter must be included as part of the prefix.

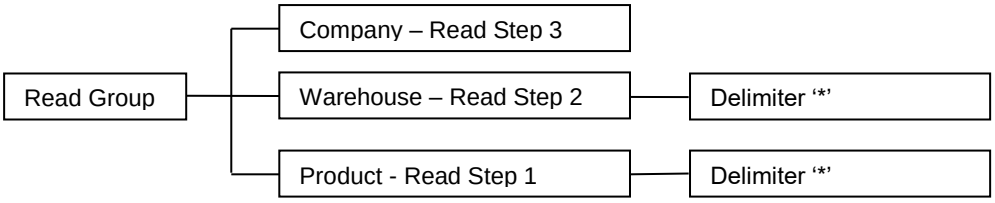
Example Key	Prefix
COMPANY*01	- the prefix is COMPANY*
STATUSREC*01*02	- the prefix must be entered as STATUSREC* even though the * delimiter is specified for the read
BUILDNUMBER|01*14567*4321	- the prefix is BUILDNUMBER|

Delimiter

Input Fields, Output Fields and MultiValue Fields

Establishes the delimiter to be used between each Read Step element of the key within a Read Group. Typically in a multi-part key the delimiter follows each key part except the final key part where no delimiter is required.

For example: Key = Product * Warehouse * Company
On the form they are in the following order.



Read to Variable

Input Fields, Output Fields and MultiValue Fields

Is used to define which DesignBais variable is used to store the results of the read.
Read Variables can only be used once on a form or form-set.

One of the following valid values must be selected from the provided dropdown list:

DBWORK, DBRECORD, DBOTHER.RECORD(1) – DBOTHER.RECORD(99)

Note that DesignBais does not know about sub forms that use read variables that have been defined elsewhere, in the main form or in other associated sub forms. The symptom of using a read variable in a sub form that has already been used elsewhere is that the read will not happen when you run the form.

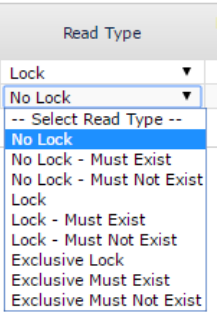
DesignBais maintains a list of all read variables and if a read is defined for a read variable that has already been defined elsewhere then the read is not added to this list and therefore the read will not happen.

The Review button in Forms Designer provides a way to display all read variables used in a set of forms. Read more here. Review (Menu Option)

File To Read — Input Fields, Output Fields and MultiValue Fields

The file to be read once the final read step in the read group has been entered and the key constructed.

Read Type — Input Fields, Output Fields and MultiValue Fields



No Lock — Will perform a read on the file without locking the record.

No Lock Must Exist
Will perform a read. If the record does not exist an error will result and the input which triggered the read will be treated as invalid.

Lock — DesignBais employs optimistic locking techniques. This means that a record lock is not maintained after the initial read. When the record is later written, DesignBais checks the record on the file to see if it is the same as the record that was originally read. If the record is different the write does not complete and an error results.

The **Lock** read type checks for a pessimistic (READU) lock condition. If one exists for the required record an error is displayed informing the user that they cannot load the record at this time.

If the intention is to have DesignBais write the record then a valid **Lock** type must be used.

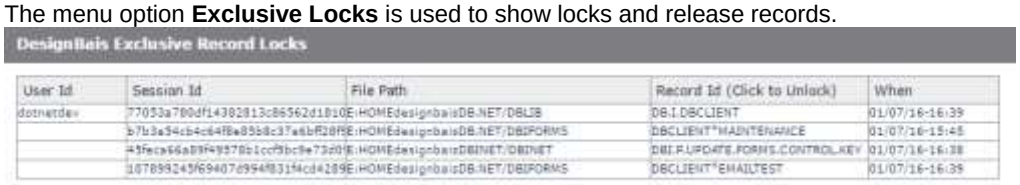
The program variables DBORIGINAL.RECORD and DBORIGINAL.OTHER.RECORD(n) store the original record data for comparison.

Lock Must Exist — Follows the same rules as the Lock mechanism. If the record does not exist on the file, an error will result and the user input which triggered the read will be treated as invalid.

Lock Must Not Exist
Follows the same rules as the Lock mechanism. If the record does exist on the file, an error will result and the user input which triggered the read will be treated as invalid.

Exclusive Lock — When a read is performed with this lock, DesignBais will record the lock in its internal lock table. This is not a pessimistic lock, but a soft-lock that DesignBais records.
When the user changes to a form that is not a child form, the lock is released. When the browser is closed the lock is released. When a write, delete or clear operation is performed, the lock is also released.

The menu option **Exclusive Locks** is used to show locks and release records.



The DesignBais editor adds a lock to the DesignBais lock list. When you are finished editing a record in the DesignBais editor you must use the File>Close (Alt-Ctrl-C) option to release the lock.

Exclusive locks are recorded in DBISESSIONS records with record ids of the form *L|weblogon*.

Code editor creates an additional lock in DBISESSIONS with id of *EDITOR*checksum.*
For example: "EDITOR*2456" where 2456 is the checksum of the path to the locked record id.
The full path to the record is held in attribute 6. All attributes are multivalued.
<1>: 19857 The date the lock was set
<2>: 66928.797 The time the lock was set
<3>: legj The user record that holds the lock
<4>: DB.NET The name of the account that the locked record is in

<5>: 1 The number of locked records held by this checksum
<6>: E:\HOME\DESIGNBAIS\DBINET\DBINET*DBI.G.RESETLNET

Exclusive Lock Must Exist

Follows the same rules as the Exclusive Lock mechanism. If the record does not exist on the file, an error will result and the user input which triggered the read will be treated as invalid.

Exclusive Lock Must Not Exist

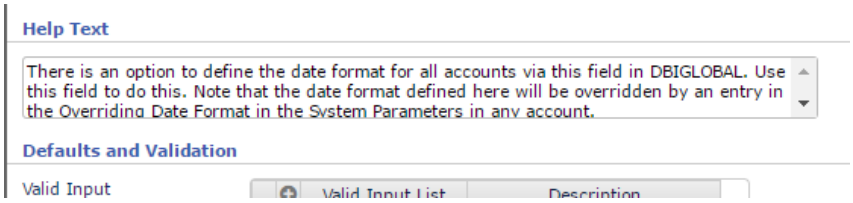
Follows the same rules as the Exclusive Lock mechanism. If the record does exist on the file, an error will result and the user input which triggered the read will be treated as invalid.

View Field Properties

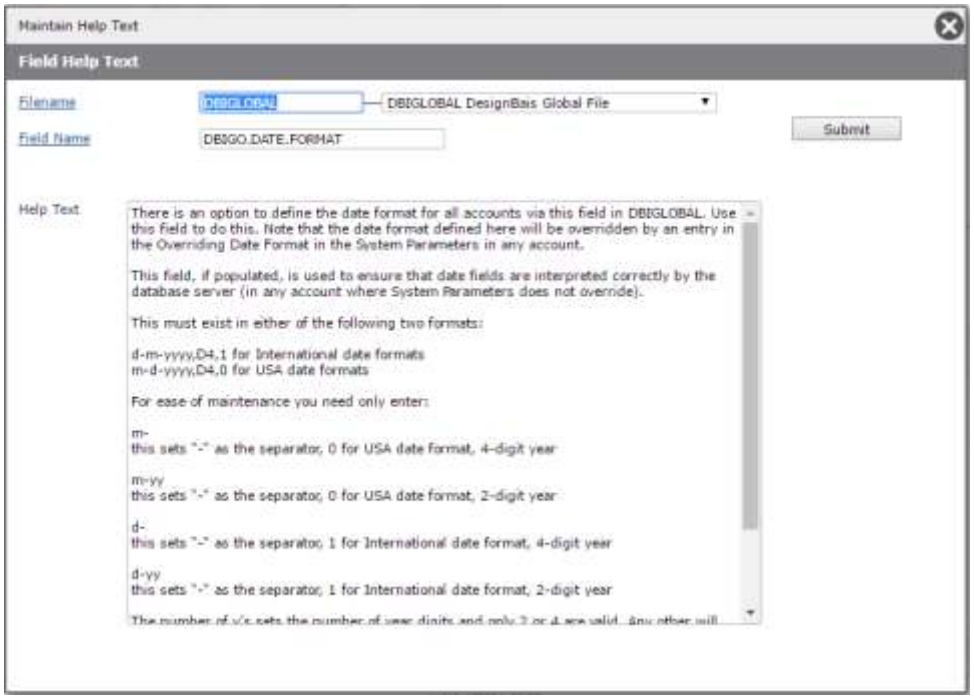
Click this hyperlink button to view the field properties of the form field element displayed in the Field Definition form. You will then see the

View Help Text

The Help Text can be viewed and amended in the Field Properties form mentioned above but this button opens a form that allows a better view and easier edit of the help text. The following example of the help text for the DBIGLOBAL field DBIGO.DATE.FORMAT demonstrates this.



Above is the view available in the Field Properties form. Below is the much easier to read and maintain version displayed via the 'View Help Text' button.



Form Loading and Subroutine Calls

Convention

In any of the 'Process' fields, the developer may enter either a valid form name or a subroutine name. DesignBais first checks if the value entered in a 'Process' field is a form resident in the DBIFORMS file (format is FILENAME_FORMNAME). If not, it is assumed to be a subroutine. In this instance DesignBais checks for a catalogued version of the program. If the catalogue does not exist, an error results and the input is treated as invalid.

Beginning in version 4.1, the code "C:" may also be used to enter a Code Block. Code Blocks allow form-specific validation code to entered and compiled directly from the form in which it is used. For some developers this may be preferable to putting event handling code in large programs where CASE statements determine the code that will be executed for specific forms, fields, and events.

Process Before

Non-Multivalue Input Fields

Will be invoked when an input field gains focus.
PROCESS.EVENT will be set to "BEFORE FIELD".
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information.

Excessive use of Before and After logic creates additional network traffic and affects the perceived performance of the application. Use these events judiciously.

Process After

Input and On-form Report fields

Will be invoked only when an input field's value changes, not if the focus changes without data changes. Caution should be used with MultiValue fields where other events are specified in Multivalue Control Properties.
PROCESS.EVENT will be set to "VALIDATE".
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information.

An input field must have a Process After if the field is duplicated on the form. Each occurrence of the field must cause a server hit in order for the value entered into any particular occurrence of the field to be duplicated into all other occurrences of the field. The process can be a dummy process that does no processing as long as the server hit is generated. A radio button field, however, cannot be duplicated.

A process after event may be required to refresh the browser with the correct value for a field. This can be a "dummy" process in that it is only required to achieve a server hit. The effect of the dummy server hit is to align SCREENREC (DesignBais common variable) with the current values in DBRECORD or DBWORK. When you encounter an issue where a field value does not refresh correctly the first step to try is to place a "dummy" process on the field.

The following clarifies what is happening in this scenario. The entered field value is returned from the web with, say, a button click but does not get into SCREENREC until after the button logic has been processed. This means that the "new" value from the logic matches the existing value and does not trigger a change for the web return. This order of processing is so that validation events can reject the entered value.

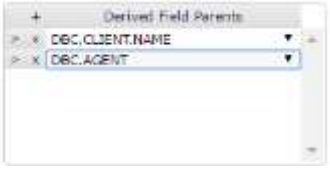
Derived in Subroutine

Input and Output Fields.

A Derived field's value will be calculated when a Derived Field Parent's value changes. Each Parent can be considered a trigger, which executes the Derived Subroutine as part of the Parent's Process After activity.

PROCESS.EVENT will be set to "DERIVED".
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information.

In order for this to function correctly, each Derived Field will need to have parent fields defined. When you enter a valid Subroutine name in this prompt (or for Code Block), a multivalue field is displayed that allows the developer to select the Parent Fields. A Derived field may have multiple parents.



Derived fields are calculated after all reads and validations are complete. This means that all of the fields required are in their respective, DBRECORD, DBOTHER.RECORD or DBWORK variables. Derived Fields alleviate the requirements for After Processes on multiple fields and the tracking of DBVALUE for derived/calculated fields.

Click event on process After

MultiValue Input Fields

Will call the subroutine or form in the Process After field when the cell is clicked, not when the cell is changed. Fields that have this flag checked cannot be modified. Will be invoked when a read occurs as a result of a change of the field's value.

PROCESS.EVENT will be set to "VALIDATE" for an input field, or 'BUTTON' for an output field.
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information.

A special case is available from Release 8.3.1.2. Setting the *Custom Attributes* of the field to a value of *readonly* allows the field to be modified by a Header Process. The presence of the custom attribute allows the user to tab into the field. The user cannot click the field to gain focus since this will run the click event associated with the field. When a particular row of the field has focus then a value from the Header Process can be placed into the field. PROCESS.EVENT is set to 'BUTTON'.

A textarea in a grid is a special case as the user needs to be able to scroll the text if it is longer than the grid row can display. This means that it is flagged as active but gets a readonly attribute.

DBSECTIONSPEC and DBENABLEFIELD can be used to enable fields, or all fields in a section. To prevent the field becoming active add a custom attribute *readonly="readonly"* which is detected by DesignBais and leaves the field as read only.

Change Event Will Fire - On Loss Of Focus

Typically the change event for a field is fired only when the value within the field changes. If this check box is set to true (checked), the change event will fire when the field loses focus, even if the data has not changed. This feature should not be used for all fields. It should be used sparingly as it will have a detrimental effect on performance. Refer to note **Change Event Will Fire** above. Note that this option cannot be selected if the field is flagged as *Mandatory*.

Process Before Read

All Input and Output fields

Will be invoked before a read occurs. This will enable the developer to change key values before a read takes place.

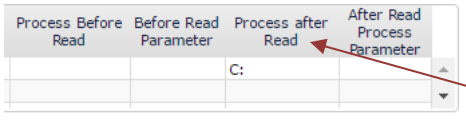
PROCESS.EVENT will be set to "BEFORE READ".
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information

Process After Read

All Input and Output fields.

Will be invoked after a read occurs as a result of a change of the field's value.

PROCESS.EVENT will be set to "AFTER READ".



If using a Code Block set focus in a cell with a C:, then click the header link to modify the code block.

Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information

Header (Hdr) Process

MultiValue Input and Output Fields

Will create a click-able header in a Multivalue field. The description in the header will be treated as a hyperlink.

Example of a Multivalue field with a Header Process attached.

Filename	Connective	Dictionary
	WITH	

PROCESS.EVENT will be set to “MV_HEADER”.
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information

Parameter

Input and Output fields

Can be used by the developer to provide extra information to a subroutine being called in one of the ‘Process’ slots. The parameter being set will be placed in the variable PROCESS.PARAMETER.
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information

Button - Specific Properties

Button Name — Buttons Only

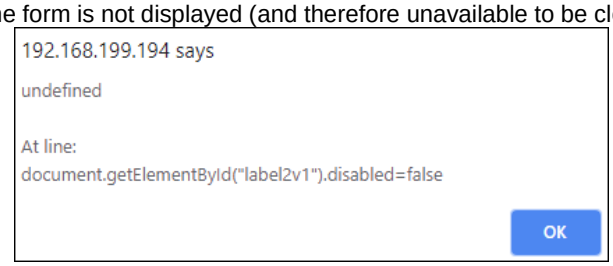
Buttons are referenced by names beginning with "B." This naming convention is required so that other processes can find all of the buttons on the form.

Field Text — Buttons and Text Only Fields

The text which is displayed on the button. This is not the same as the Button Name.

Close Modal Window — Buttons Only

When the button is pressed, a modal form will be closed. If a 'Process After' is also being used when a button is pressed the Close Modal Window property is ignored. You must use PROCESS.CLOSE.MODAL = 1 (in a subroutine) to close a modal window in this instance. Note that if you set PROCESS.CLOSE.MODAL = 1 when the form is not displayed (and therefore unavailable to be closed) then you may see undefined errors:



Note that if the *Confirm Update* option is set for a submit or delete button then the button click raises the confirm dialog but also closes a modal form. In this case it is necessary to deselect the *Close Modal Window* check box and set PROCESS.CLOSE.MODAL = 1 in a subroutine.

Return to Field — There are two different functions for this field. The function depends on the process being invoked.

Calling a search form:

> When calling a search form the Return to Field prompt must be filled with a field name that appears on the form. This will ensure that the search process returns the result (from the search) to the correct field. Search forms may be invoked directly with a Process After as FILENAME_SELNAME, or a selection process may be launched from a program specified in the Process After field, using the PROCESS.STACK variable.

Button Process (Updates and Clear):

> Setting the Return to Field on a button to a valid field name will position the cursor (place focus) on the nominated field.

Check Box - Specific Properties

Value if True — Check Box Only

Sets a value to be returned to the database if the Check Box is checked.

Value if False — Check Box Only

Sets a value to be returned to the database if the Check Box is unchecked.

Group Name — Is used to group more than one Check Box together. If one of Check Boxes in the group is checked the others are unchecked. This simulates the behaviour of radio buttons. This cannot be used in conjunction with the 'Process After'

Image – Specific Properties

Image File

Image Only

Defines the name of an image to display on the form. The default is the DesignBais logo image, db/dblogo.jpg.

Also, when used in code, the Image File is not necessarily the Image filename but a reference for the unique object on the form. For example:

DBIMAGESPEC<1> = “LOGO1”
DBIMAGESPEC<2> = “MyCompLogo.jpg”

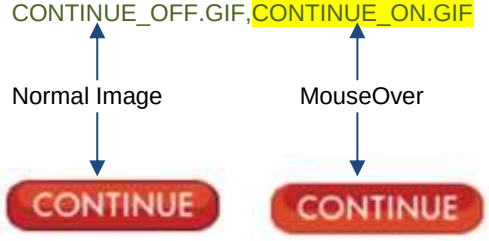
In this case, LOGO1 is not a valid image file name, and will not display a real image in development mode. Referring to an image object like LOGO1 is much more intuitive than referring to it as the default DBLOGO.JPG, especially when there are several images on the form which need to be managed individually at run-time. This name is also returned as the PROCESS.EVENTSOURCE for IMAGE events.

Application images must be loaded into the images directory, which is a child to the DesignBais site physical directory (DBNET). It is recommended that you create a sub-directory under the existing images directory to store images. Eg. Images\myimages. If this is the case the image name entered for this property would be myimages\imagename.ext. (Either a forward or backward slash is valid.)

Note that the image column span can be omitted. Images are then scaled by setting the Row span for the image height in pixels, and the image width is scaled in proportion to the actual image size.

There can be two image names entered in this field (separated by commas). The first image name is the normal state; the second image listed is the focus state.

Example: CONTINUE_OFF.GIF,CONTINUE_ON.GIF



Display Class

A display class can be entered. This would allow, for example, an oval border to be placed around an image.

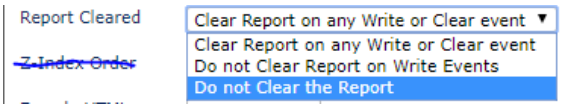
On-form report - Specific Properties

Report Name

The name used to refer to a specific On-form Report in code. By convention OnForm Report Names begin with “R.”. Eg, “R.SALES”

Report Cleared

There are 3 options as shown in the figure.
Clear Report on any Write or Clear event is the default.
The 2nd option allows you to clear the report on a clear event but not when a write is done.
The last option means that the report remains populated after any write or clear event.



Report Number

A form can have up to ten different On-form Reports displayed. Each report must be provided a unique number enabling DesignBais to update the correct report when instructed. If the report number is left blank, then DesignBais assumes that the report number is one (1). Reports and Graphs share report buffers, therefore the Report Number must be unique among all reports and graphs on a form. **Graphs are not available in Version 7 and later releases**.

Graphs - Specific Properties

Field Name

The name used to refer to a specific Graph in code. By convention Graph Field Names begin with “G.”. Eg, “G.SALES”

Report Number

A form can have up to ten different graphs displayed. Each graph must be provided a unique number enabling DesignBais to update the correct graph when instructed. If the report number is left blank, then DesignBais assumes that the graph number is one (1) Reports and Graphs share report buffers, therefore the Report Number must be unique among all reports and graphs on a form.

Multivalues (Menu Option)

This menu option displays the Multivalue List or control form. The fields in this form are used to describe the Multivalue controls that appear on the form, the depth of each control (height of the container in pixels) and what processes (subroutines only) to call (if required) when an end-user clicks on one of the row control buttons.

A grid is a group of related multivalued attributes. Each column in the grid represents an attribute or field, and each row represents the same value for each of the fields. In the above example there are three multivalue grid names displayed. This indicates that there are three multivalue association grids on the form that is being designed.

Typically multivalue association grids are named ASSOC*n.* The ASSOC indicating a grid and the *number (n)* indicates what grid the name is representing.

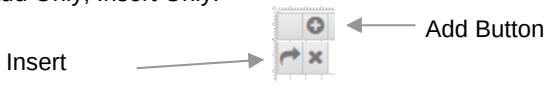
In the above example, the first grid is named INVLIST1. This indicates (a name other than ASSOC) that all fields within the association grid have been flagged as belonging to a group (*refer to the Field Properties chapter in this manual*). In this scenario, any field that belongs to the group INVLIST, will be maintained whenever a row is added, inserted or deleted whether it is included in the Multivalue grid or not.

If the example below, there are three fields from the group CONTACT added to the form. There are two other fields, Contact Phone and Contact Fax which have been flagged for the CONTACT group, but these have not been added to the form. Whenever a row is added, inserted or deleted in the control below, Contact Phone and Contact Fax will also have a row added, inserted or deleted.

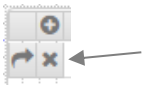
Example Multivalue Control

Prompts

Group	The name assigned to each Multivalue grid control on the form. This field cannot be modified. DesignBais automatically appends a number to the end of the group name specified in Field Properties.
Depth	The depth (height) of the Multivalue grid control (in pixels) on the form. The vertical scrollbar becomes active when more rows than will fit within this range are present.
Add/Insert Allowed	Will enable the Add and Insert buttons in a Multivalue control. Valid options are: Allowed, Not Allowed, Add Only, Insert Only.



Delete Allowed — Will enable the delete button in a Multivalue control. Valid options are: Allowed, Not Allowed, With Dialog.



Add Process — Is used to define a subroutine to call whenever the Add button is pressed in the Multivalue control.
PROCESS.EVENT will be set to “MV_ADD”.
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information.

Entering C: in this field indicates that a code block will be used. Set focus in a cell with a C:, then click the header link to modify the code block.

Process Before Insert — Is used to define a subroutine to call whenever the insert button is pressed in the Multivalue control.
PROCESS.EVENT will be set to “MV_INSERT”.
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information.

Entering C: in this field indicates that a code block will be used. Set focus in a cell with a C:, then click the header link to modify the code block.

Process Before Delete — Is used to define a subroutine to call whenever the delete button is pressed in the Multivalue control.
PROCESS.EVENT will be set to “MV_DELETE”.
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information.

Entering C: in this field indicates that a code block will be used. Set focus in a cell with a C:, then click the header link to modify the code block.

Parameter — Can be used by the developer to provide extra information to a subroutine being called in one of the ‘Process’ slots. The parameter being set will be placed in the variable PROCESS.PARAMETER.

Post Action — After an Insert, Delete or Add action, the Post Action may be used to perform additional processing on the row that has had the action. This may be useful to load default field values in a multi-value row after an Add or an Insert has been performed.

PROCESS.EVENT will be set to "MV_POST"
PROCESS.EVENTSOURCE is the name of the association group.
PROCESS.PARAMETER - Please see the following Parameter field.

Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information about the events and variables related to these controls.

Example - Setting default row value to the value from the preceding row (taken from the DesignBais Demo Form DBDEMO_GRID.DEFAULT:

```
MV.POST:
      BEGIN CASE
         CASE SCREEN.NO = “GRID.DEFAULT” AND (PROCESS.PARAMETER=”MV_ADD” OR PROCESS.PARAMETER=”MV_INSERT)
            IF DBMVCOUNT > 1 THEN
               DBRECORD<DEM.INV.NUM,DBMVCOUNT> = DBRECORD<DEM.INV.NUM,DBMVCOUNT-1>
            END ELSE
               IF PROCESS.PARAMETER="MV_INSERT" THEN
                  IF DBRECORD<DEM.INV.NUM,DBMVCOUNT+1> # '' THEN
                     DBRECORD<DEM.INV.NUM,DBMVCOUNT> = DBRECORD<DEM.INV.NUM,DBMVCOUNT+1>
                  END
               END
            END
      END CASE
      RETURN
```

Entering C: in the Process or Post Action fields defined below indicates that a code block will be used. Set focus in a cell with a C:, then click the header link to modify the code block.

Parameter (Post Action)

A parameter that may be used to support to Post Action. This can be any value. If the value of the parameter is null, DesignBais will place the original action into the Parameter. I.e.: MV_ADD, MV_INSERT, MV_DELETE. This will make it much easier to determine the reason for the post action call.

Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information about the events and variables related to these controls.

Entering C: in the Process or Post Action fields defined below indicates that a code block will be used. Set focus in a cell with a C:, then click the header link to modify the code block.

Suppress Scrollbar

Enter "Yes" if the vertical scrollbar for the multivalue grid is to be suppressed. Suppressing the scrollbar increases the space available for the grid columns by the width of the space reserved for the scrollbar. If the maximum number of grid rows is known and the grid is sized to display them, then suppression of the scrollbar is a good option.
Setting “Yes” to Suppress Scrollbar is the equivalent of setting DBMVPROP<11> = 1 (refer to Styling a Multivalue Grid Programmatically). If either are set then the 18px space allowed for the scrollbar is available to the data columns of the grid.

Controlling Attribute for a Multivalue Dataset

DesignBais does not require the controlling attribute of a multivalue dataset to be defined. It is often implied by the order of the fields in a multivalue grid element on a form but it is not essential to have the controlling attribute as the first field in the grid.

It is essential and therefore important that all associated attributes of a multivalue dataset have the same Group Name in the Field Properties field DBIP.ASSOC.GROUP. This allows DesignBais to maintain the correct association of values within attributes of the multivalue data set.

In assigning a value to Group Name note the following:

- A Group Name must only contain alpha and numeric characters
- Group Names must not contain a dot (.)
- If a Group Name includes the string “ASSOC” then DesignBais will NOT maintain the association of multivalues in fields that are not on the form.
- If the Field Properties of fields within a multivalue grid do not have a Group Name then DesignBais assigns a default Form Group Name of “ASSOC” followed by an integer to maintain unique names within the form.

It is recommended that Multivalue Grids normally have an input field as the first column. If this is not the case, because the developer wants to place an output field as the first column, then the *Add/Insert Allowed* and *Delete Allowed* options in the **Multivalue Control Properties** should be set to *Not Allowed*. This will prevent the user from adding, inserting or deleting rows from the grid.

Importantly it will also allow the user to tab out of the final row of the grid, rather than having to use the mouse to set focus to another form element. By default, if *Add/Insert Allowed* is set to *Allow* and the first column of a grid is not an input field or a clickable field then tabbing through the last populated row of the grid will keep adding empty rows.

Buttons

Submit

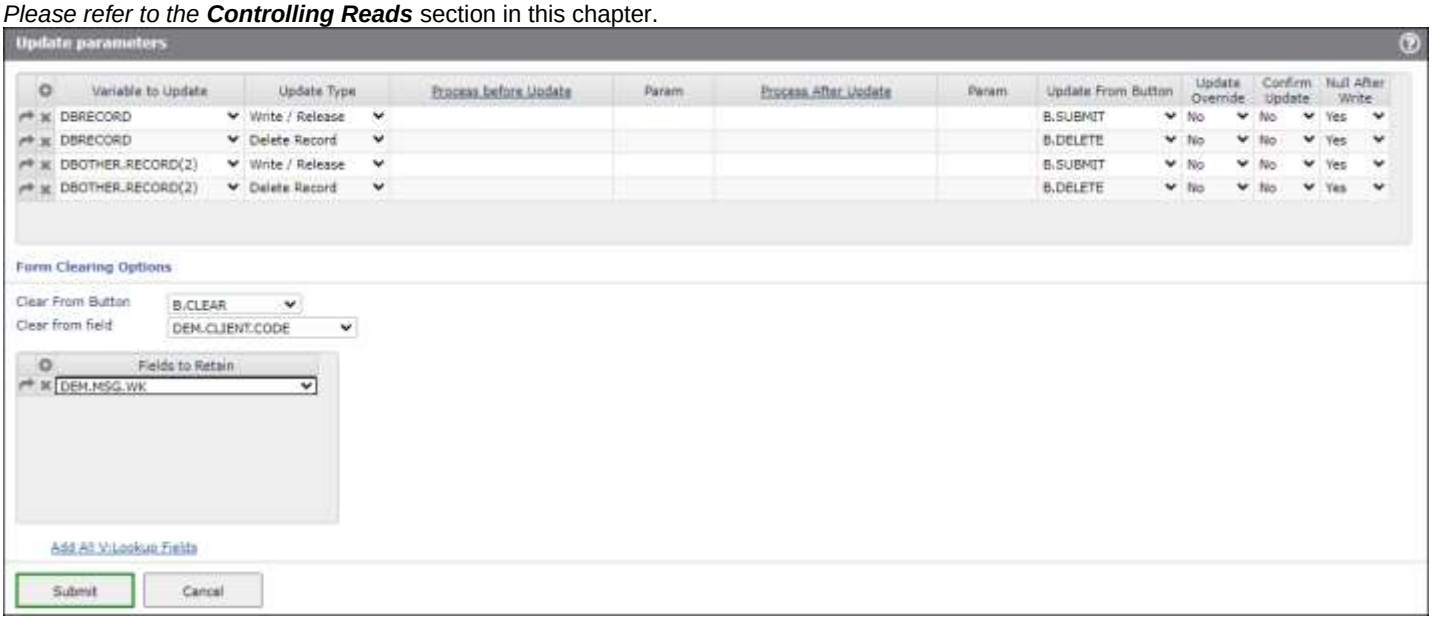
Will commit the changes made on the Multivalue List form.

Cancel

Will cancel any changes made on the Multivalue List form.

Updating (Menu Option)

This menu option is used to describe the write processing that DesignBais will perform when an assigned button is pressed. Writes will not function correctly unless the variable being written has been previously read by DesignBais using one of the Lock Read Types. *Please refer to the* ***Controlling Reads*** section in this chapter.



Prompts

Variable to Update

Select the variable name that you wish to be updated from dropdown list provided. The variable selected will be updated when the nominated button is pressed.

You may nominate multiple file updates to occur when a button is pressed, and the same file variable can be processed differently depending on which button is pressed.

If the variable has been read (by DesignBais) within a multivalue grid then the DBOTHER.RECORD(n) variable may not be written by DesignBais. This restriction is required because the read within the grid lowers the field separators (AM to VM, VM to SVM, SVM to defined string) in order to display the record attributes in the columns of the grid.

Update Type

There are three different Update Types.

Write/Release

Will write the selected record and release the optimistic record lock for the variable selected.

Write/No Release

Will write the selected record and keep the optimistic record lock intact. This allows for the design of forms that write the same record multiple times without being trapped by the optimistic record lock verification after the first write.

Delete Record

Will delete the selected record from its read file.

Process Before Update

This field is used to define a subroutine to be called before the write process. This provides for extra validation to be applied, or for data to be modified before a write.
PROCESS.EVENT will be set to “BEFORE WRITE”.
Please refer to the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information.

Entering C: in this field indicates that a code block will be used. Set focus in a cell with a C:, then click the header link to modify the code block.

During the validation phase you can interrupt a write by setting FILE.OPERATION.STATUS to true (1).

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Setting IERR.TEXT will not stop a write process. Note however that if FILE.OPERATION.STATUS is set to 1 but IERR.TEXT is left null then the write will not occur but the form may still close and a new form open if for example PROCESS.STACK is populated.

```
Eg.     IF VERIFICATION.FAILED THEN
            IERR.TEXT = "Verification Failed"
            FILE.OPERATION.STATUS = 1 ; * Abort the write process
        END
```

Process After Update

This field is used to define a subroutine to be called after the write process. This provides for additional write processing to occur after the main DesignBais record write.
PROCESS.EVENT will be set to "AFTER WRITE".
Please refer to the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information.

Entering C: in this field indicates that a code block will be used. Set focus in a cell with a C:, then click the header link to modify the code block.

Parameter

Can be used by the developer to provide extra information to a subroutine being called in one of the 'Process' slots. The parameter being set will be placed in the variable PROCESS.PARAMETER.

Update From Button

The update of the record specified will occur when the user presses the button defined in this field. There may be multiple buttons with the same name placed on a form.

Update Override

When set to 'Yes' DesignBais will not perform any writes. It is assumed that a subroutine named in the 'Process Before Update' or 'Process After Update' slots will perform the write.

Update Confirmation

When set to 'Yes' will prompt the user to confirm the write/delete process. Typically this would be used for the Delete update type.

Null After Write

When set to 'Yes' DesignBais will clear/null all of the records that were associated with the write being performed. If the record being cleared is the parent to other reads then these other reads will also be cleared.

Clear From Button

When the button named in this field is pressed, fields on the form will be cleared as defined by the setting in the Clear From Field described below.

Clear From Field

When the clear button is pressed, or null selected after an update, then fields are cleared as follows:

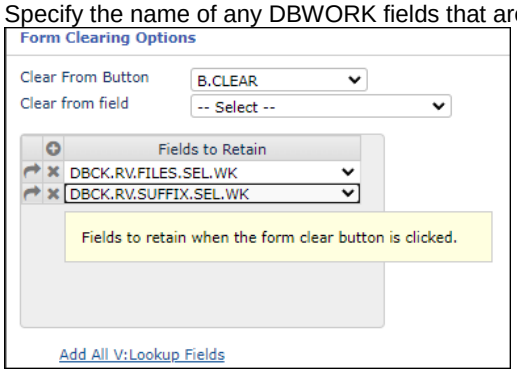
- all Read Variables and associated keys (such as DBRECORD and DBKEY, DBOTHER.RECORD(n) and DBKEYS(n).
- all DBWORK fields after the field name entered will be cleared. DBWORK fields with a tab index or row/column position less than that of the field specified here will not be cleared.

Focus will be on the field after the field specified here.

This is useful to retain partial keys in a multi-part key record. If left blank all fields will be cleared.

Fields to Retain

Specify the name of any DBWORK fields that are not to be cleared when the *Clear From Button* is clicked.



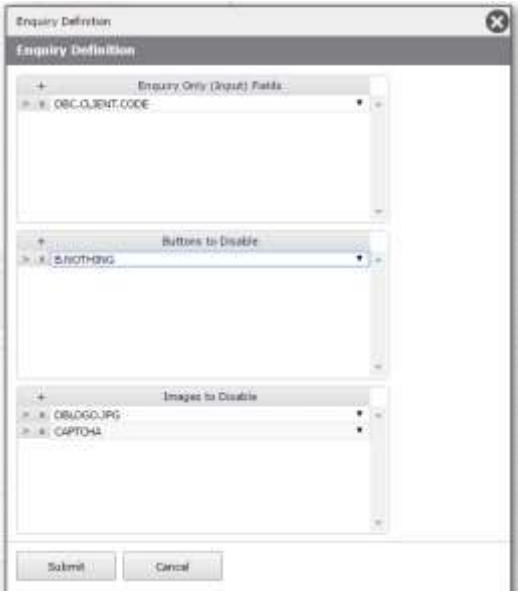
Add All V:Lookup Fields	Click this button to populate the *Fields to Retain* grid with the names of any DBWORK fields that are referenced via the *V:workFieldName* structure.

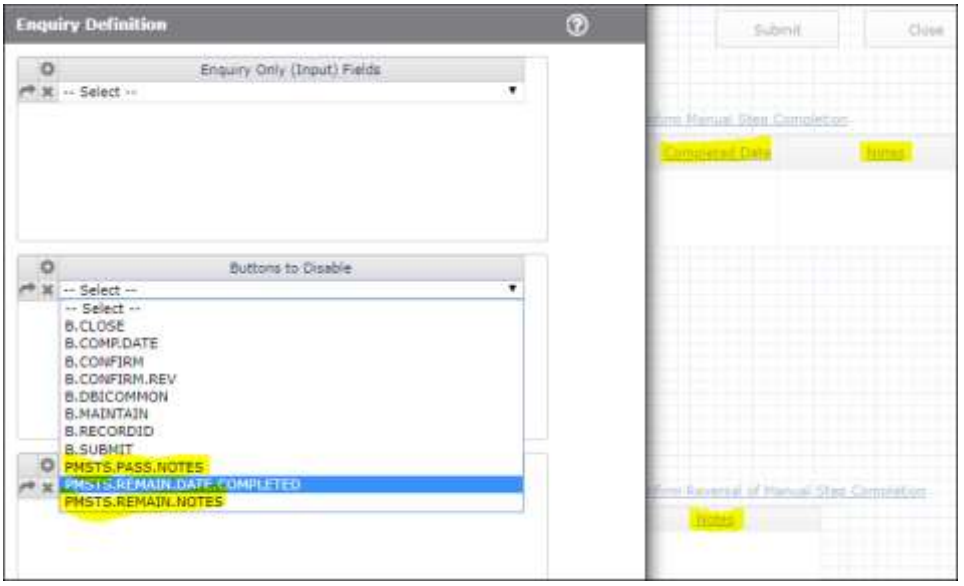
Buttons

Submit	Will commit the changes made on the Update Parameters form.
Cancel	Will cancel any changes made on the Update Parameters form.

Enquiry (Menu Option)

This menu option is used to determine which fields on the form are enabled when the form is running in enquiry mode. Typically the fields named on this form would be used for entering record keys(s) during enquiry mode. Any form can be presented for updating data, or in enquiry mode depending on the menu used, or code settings. See entries for ~E in the index for related information.





Prompts

Enquiry Only (Input) Fields — This Multivalue field is used to determine which fields on a form are enabled in enquiry mode.

Buttons to Disable — In Enquiry mode, it may be necessary to disable buttons. Enter the button names in this prompt and they will be disabled during enquiry mode. DesignBais automatically disables any buttons that have an update/delete operation attached to them, so it is not necessary to define those buttons in this field. This option also allows for the exclusion of Multi-value header click events in enquiry mode. Images with associated events are not listed here.

Images to Disable — Select the images that are to be disabled in Enquiry mode.

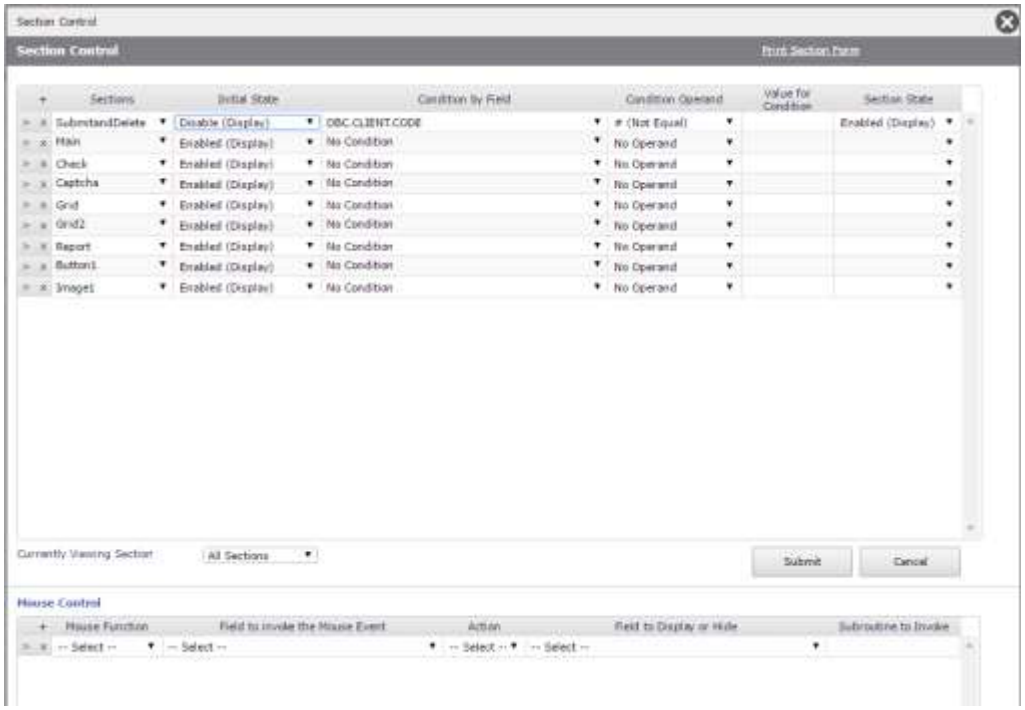
Buttons

Submit — Will commit the changes made on the Enquiry Definition form.

Cancel — Will cancel any changes made on the Enquiry Definition form.

Sections (Menu Option)

This menu option is used to define the display state of every control on the form during runtime.



This form is also used to control the display of sections in development mode. The developer has the choice of displaying, adding to or modifying controls that belong to an individual section or may display all sections.

Section display states are controlled by fields known as **Trigger Fields** which are referenced in the “Condition by Field” column of the Section Control form. Fields used in the “Condition by Field” may be input or output fields on the form, or they may be hidden on the form and only used to control the states of other sections.

In the example above the value in the field **DBC.CLIENT.CODE** is used to control the enabling of the Submit and Delete buttons both of which would have a section name of **SubmitandDelete.** The **SubmitandDelete** section is initially Disabled and therefore the Submit and Delete buttons are only enabled when a value is entered in the field DBC.CLIENT.CODE. Note that the “Value For Condition” is empty such that the condition reads IF DBC.CLIENT.CODE # “” THEN enable fields in the SubmitandDelete section.

These trigger lines can be read in a few ways. For example: “the initial state is ‘this’, but when the indicated variable matches the specified condition, change the state to ‘that’. Another way to read it is “the state is ‘this’ unless or until the indicated condition is true, then reset to ‘that’. Given this logic, the same section can be listed more than once, with different conditions triggering different behaviours.

Section control logic is reviewed on each server event to determine if some user action has caused changes to the state of the form.

Sections can also be controlled by the Variable DBSECTIONSPEC. *Please see the Common Variables and Subroutines Section in this document for further details*

Prior to Version 7 the developer could be enter section names in the Section Control form and there was no validation, and section names would remain in the list even if no form elements were assigned to the section. Developers also had to manage the sorting of section names such that form collapsing functioned correctly.

With Version 7 and later this has changed. Section names can only appear in the Section Control form if they are entered in the Section property when adding or maintaining form elements. A null section name will be set to the default section name ‘Main’. The Section Control form will display section names in row and column sequence such that form section collapsing, if required, will function correctly. Existing section names can be inserted in the Section Control Form if required but this is controlled by forcing selection of an existing section name from a dropdown list. Section and Subsection names must be entered on a form element in order for these names to appear in the dropdown list.

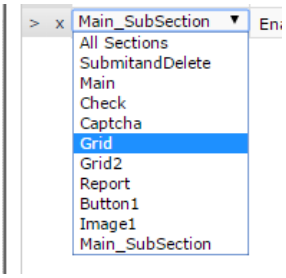
Subsection names will be appended to the list of section names in the Section Control form since these must appear after all other section names.

Section and Subsection names can appear in the Section Control form Sections list multiple times where this is required in order to define multiple conditions to control the enabling, disabling or hiding of form elements.

Prompts

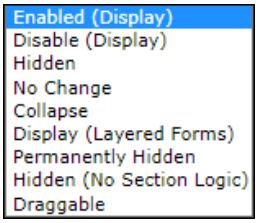
Sections

Lists the names of the Sections and Subsections that have been added to the form.
The same Section or Subsection name may be listed more than once to create more refined behaviours. Select from the dropdown list.



Initial State

Initial state determines the initial display state for all controls belonging to the named section.



Enabled (Display)
The option value 'ENABLE' indicates that the initial state of all controls belonging to a section will be enabled
and displayed.

Disable (Display)
The option value 'DISABLE' indicates that the initial state of all controls belonging to a section is disabled.

Hidden
The option value 'HIDDEN' indicates that the initial state of all controls belonging to a section is hidden.
Note: By convention, fields which are intended to always be hidden are usually put into a section named 'Hidden'.

No Change
The option value 'NOCHANGE' indicates that an initial state of all controls belonging to a section will not be set. The display state of these controls will be derived from the condition-based section state prompt. Most sections are assigned a specific initial state. When refining the state of a section with multiple control lines, the initial state should not be reset, as doing so will invalidate prior instructions. The No Change state generally assumes other changes have been made, and that no change in state will be made unless the defined condition is true. More information is provided for this setting in the next section.

Collapse
The option value 'COLLAPSE' indicates that the section will be collapsed until a certain condition applies. *See the section on collapsing sections later in this chapter.* Note that a section that is to collapse MUST have an initial state of collapse.

Display (Layered Forms)
The option value 'LAYER' indicates that the section will only be displayed when the form is loaded in a layered state (Process call has '~L' appended). This is useful for adding Close buttons to a form that are not visible in normal mode, but are visible when the form is layered.

Permanently Hidden
The option value 'PHIDDEN' functions in the same way as Hidden but in addition reduces traffic to the server. DesignBais does not send any changes to and from the server for form elements that are permanently hidden. Conditions cannot be entered if the initial state is Permanently Hidden. Permanently Hidden fields are not sent to the web at all. Section control fields can be flagged in this fashion to cut traffic. The data is available in the database component to make decisions about sections to be displayed.

Hidden (No Section Logic)
The option value 'IHIDDEN' indicates that the initial state of all controls belonging to a section is hidden.

Draggable

The option value ‘DRAG’ indicates that the controls in this section can be dragged using the mouse..

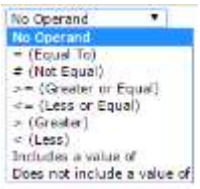
Condition by Variable

Provides the ability to display fields belonging to a section depending of the state of another field on the form.

In the pictured example the section control system would evaluate the display status of a section based on the value in the field DBC.CLIENT.CODE. Note that any field on the form can affect any section – the list of variables is not restricted to the fields assigned to the current section.

Condition Operand

Describes the arithmetic operator for the ‘Condition by Variable’ field. If set to anything other than No Operand, the condition will be based on the operator chosen and the ‘Value for Condition Field’.



The ‘Includes a value of’ option allows a single variable to control behaviour for a number of sections. For example, rather than having one work variable called WK.DISPLAY.ADDRESS and another called WK.DISPLAY.CONTACTS, and using each variable to control the display of sections Address and Contacts, a single variable called WK.CONDITIONS can be used. This variable might include the letter “A” to display the Address section, and/or the letter “C” to display contacts.
So the condition for Address would read:
“For section Address, the initial state is disabled unless WK.CONDITIONS Includes a value of A, then the section state is Enabled.”
Similarly, the condition for Contacts would read:
“For section Contacts, the initial state is disabled unless WK.CONDITIONS Includes a value of C, then the section state is Enabled.”

Value for Condition

The value entered is used to determine whether the condition is true or false. If the value equates to true then the section display parameters are changed in accordance to the ‘Section State’ field.

If null is the required entry (as in # “”) leave the field blank. Do not place “” in the field.

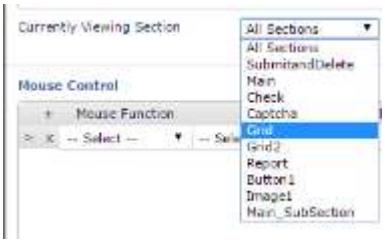
Section State

The ‘Section State’ is activated when the results of the condition are set to true.

If the result of the condition is false then the initial state is used (except when the initial state is ‘No Change’)

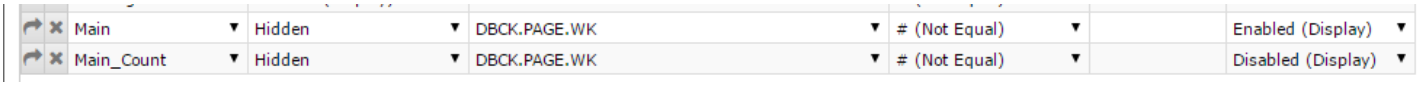
Currently Viewing Section

When designing forms with many sections it is sometimes useful to display a single section at a time in designer mode, particularly when there is a need to overlay fields for display purposes. This field allows the designer to view and develop a single section or view all sections at one time.
When viewing only a single section, functions like, Insert Row and Delete Row only act on the fields within the section. All fields that do not belong to the currently viewed sections remain unaffected.



Section Control and Disabled Fields

Setting the *disabled* property of a field on a form where the field is in a section that is controlled by the Section Control logic may require the establishment of a Subsection. In the following example a field in the “Main” section is to be disabled. The “Main” section is initially *hidden* and then *enabled* after the DBCK.PAGE.WK field is populated. As well as setting the Field Disabled property of the field it is also necessary to move the field to a Subsection (in this example “Main_Count”) in order for the section control logic to define an additional condition for just this field. The Section State becomes *disabled* rather than *enabled* after DBCK.PAGE.WK is entered.



Refer to the common variable DBENABLEFIELD for additional remarks.

The 'No Change' Initial State

The **'No Change'** Initial State provides the ability to create CASE-like statements within the section control.

Sections	Initial State	Condition by Field	Condition Operand	Value for Condition	Section State
SubmitandDelete	Enabled (Display)	No Condition	No Operand		
Main	Enabled (Display)	DBC.CLIENT.CODE	# (Not Equal)		Enabled (Display)
Check	No Change	DBC.PHONE	= (Equal To)	P	Enabled (Display)
Check	No Change	DBC.PHONE	= (Equal To)	M	Disabled (Display)
Check	No Change	DBC.PHONE	= (Equal To)		Hidden

In the above example there are three states for the section named **Check**. This provides a CASE-like condition. As there is no initial display state, there must always be a condition that results in a true state. In the above example the = NULL condition does this.

The result is a condition that looks like the following:

```
CASE DBC.PHONE = "P"
        Enable and Display the section
CASE DBC.PHONE = "M"
        Disable and display the section
CASE DBC.PHONE = ""                    ; * should be a condition that always equates to true.
        Hide the section
```

This combination of section state provides a high degree of flexibility in the display of sections on a form.

The following example is helpful to further understand the use of the *No Change* option.

The *Build* section below is set initially as *Hidden*. Then exception conditions follow that determine when to enable the section.

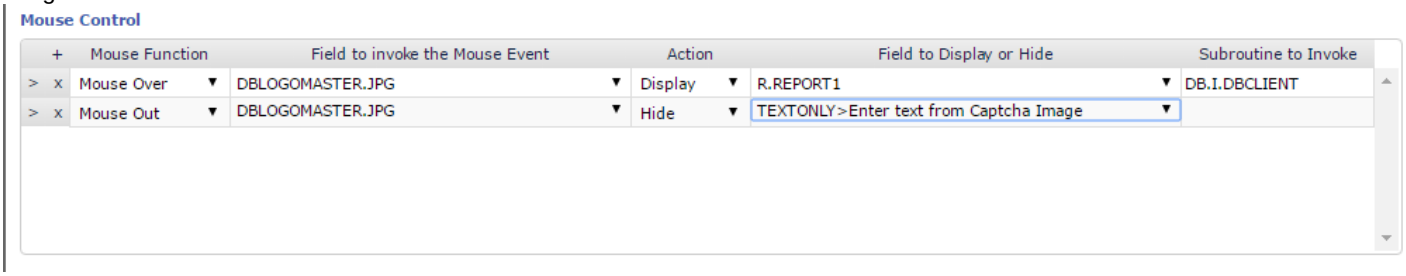
	Build	No Change	No Condition	No Operand		Hidden
	Build	No Change	DBIU.USER.ID.WK	= (Equal To)	garb	Enabled (Display)
	Build	No Change	DBIU.USER.ID.WK	= (Equal To)	legj	Enabled (Display)

The following example does NOT work even though it may appear to be a reasonable way of achieving the required action.

	Build	Hidden	DBIU.USER.ID.WK	= (Equal To)	garb	Enabled (Display)
	Build	Hidden	DBIU.USER.ID.WK	= (Equal To)	legj	Enabled (Display)

Mouse Control

Mouse control allows the designer to display and hide images, buttons and on-form reports as the end user mouses over other images and buttons.



Mouse Function — The Mouseover event has two states. MouseOver and MouseOut. This prompt defines whether the event fires during the MouseOver or the MouseOut operation.

Field to invoke the Mouse Event — When the user mouses over the nominated image or button the event will trigger.

Action — Defines what is to occur when the mouse event is triggered. The options are Display or Hide.

Field to Display or Hide	Defines what image, on-form report or button will be displayed or hidden as a result of the mouseover or mouseout event. In the example shown above, on the first line, when the user Mouses Over the image DBLOGOMASTER.JPG the on-form report R.REPORT1 will be displayed. On the second line when the user Mouses Out of the DBLOGOMASTER.JPG image the text field containing 'Enter text from Captcha Image' will be hidden. Note that the dropdown selection list for this field will contain a additional entry "rdesignbaisdrop" if the form has "Include a report for keypress searches" checked. This entry relates to the predictive text search results window and is made available in this dropdown so that it can be, for example, hidden when a mouseout event occurs. Refer to the HEADER Form Overlay section of the manual.
Subroutine to Invoke	When a user hovers over an image, there is an option to invoke a server event. The subroutine to invoke is nominated in the Section Control form in the Mouse Control Section. The event type is as follows: PROCESS.EVENT = "MOUSE OVER" PROCESS.EVENTSOURCE = *ImageName.ext*

Buttons

Submit	Commit the changes made on the form and update the designer view if required.

Collapsing Sections

Sections on a form may be made to collapse when a trigger field (another field on the form that a section condition is linked to) has a change of value. For collapsing to function correctly each section named in a collapsible form must occupy a separate contiguous region. A section that is to collapse MUST have an initial state of collapse in order to flag the section in the list of collapsing sections in DBCOLLAPSELIST.

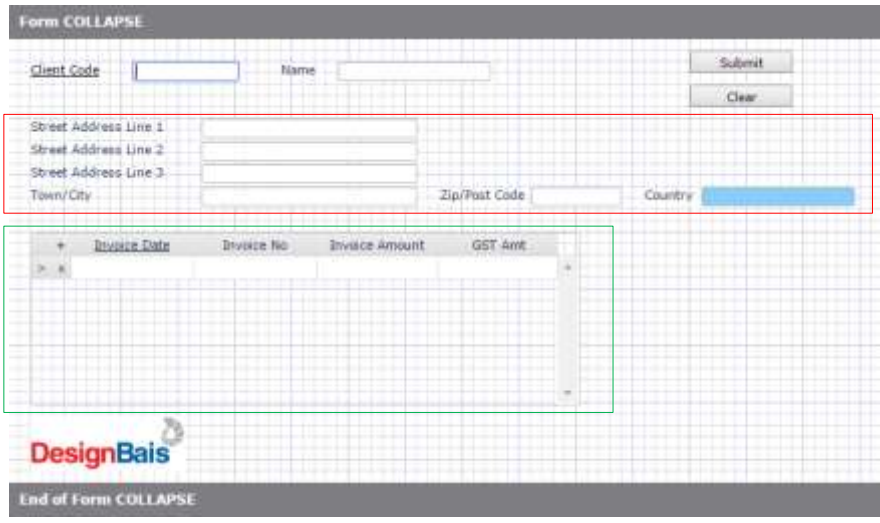
DesignBais maintains section names, in the Section Control list, in the row and column sequence of the fields on the form. The developer must ensure that each section that is to collapse contains only fields with that section name. Providing this is done DesignBais will correctly sort the section names such that collapsing will function correctly.

A Section with an Initial State of 'Collapse' can only have one entry (rule) in the Section Control list if it is to collapse correctly.

This example demonstrates how to set up section rules to create collapsing sections.

The address fields are all in a section called 'Address'.
Country is in a subsection called 'Address_Country'.

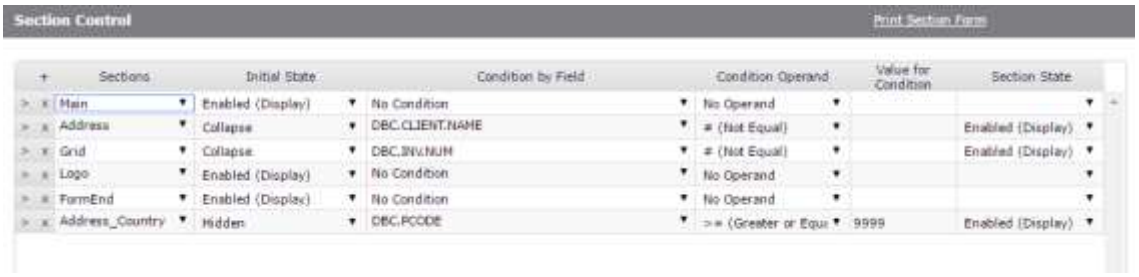
The invoice detail fields are all in a section called 'Grid'.



The Section Control rules are as follows:

- The Address section remains collapsed until the client Name is entered.

- The Grid section remains collapsed unless there is a value in the Invoice No field (DBC.INV.NUM). The invoice details can be assumed to be entered in another form.
- Within the Address section, however, the country field remains hidden unless the postcode field DBC.PCODE has a value greater than 9999. This requires a subsection which is shown.

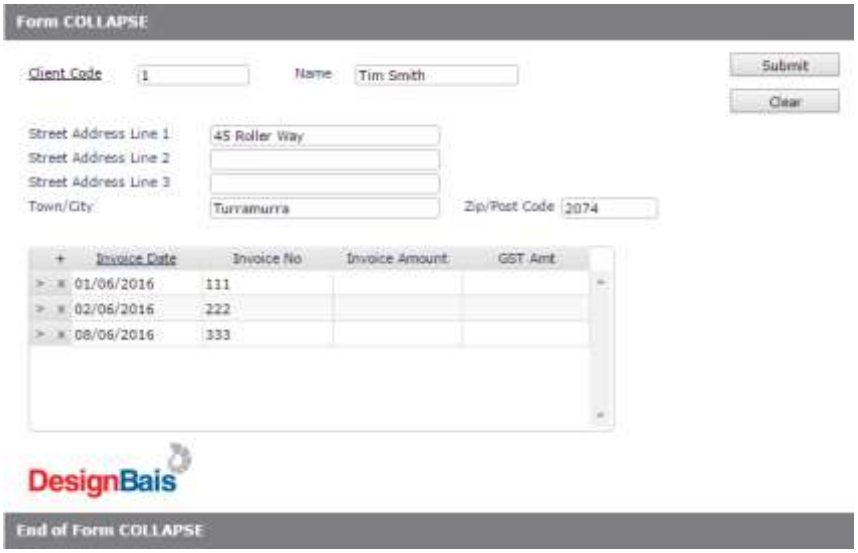


Initial state of the form:

Entry of Client code and Name un-collapses the address fields:

Entry of Zip Code greater then 9999 un-hides the Country field:

Entry of a client code record with invoice details un-collapses the Grid section:



Subsections

There may be occasions where you require fields that are placed within a section to have additional, disable, enable or hide functionality. If this is the case you can add these fields to the form with a section name comprised of two components. Each component separated by an underscore. This is referred to as a Subsection. The first component of the name must be the name of an existing section, referred to as the the parent section.

Subsections are only **required** if the parent section is required to collapse. They may be defined for other non-collapsing sections but normal section rules can achieve the required result.

If collapsing is to function correctly the row number of a Subsection must be equal to or greater than the lowest row number of any element in the parent section. In other words a subsection cannot be above the start of the parent section.

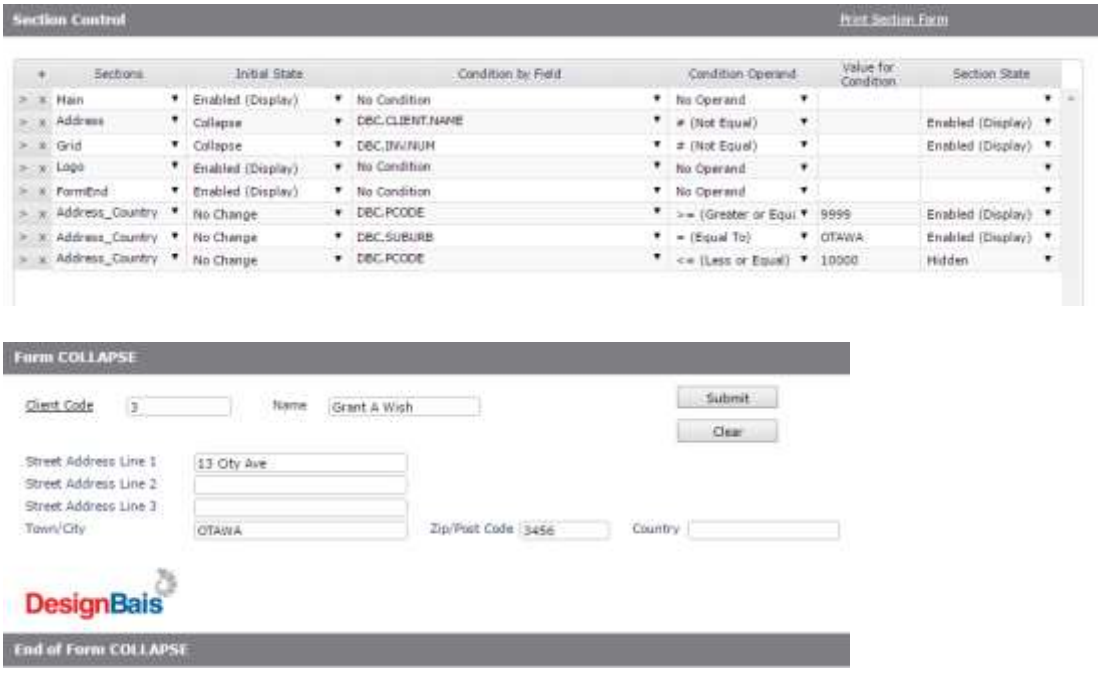
Subsections for collapsing parent sections will sort to the end of the list of sections in the Section Control form.

If the parent section is initially collapsed then the subsection will be initially hidden. When the parent is enabled then section rules for the subsection must be in place to control the display of the subsection. Alternatively, DBSECTIONSPEC or DBENABLEFIELD may be utilised.

This example demonstrates how subsections can be used to create a set of ‘CASE’ conditions. To achieve this we must change the initial state of the single subsection condition from above (where it was ‘Hidden’) to ‘No Change’. Then we can add further conditions in order to refine how the fields with this subsection name are controlled. The sequence of these conditions is important since the first condition that is true will determine the state of the subsection.

So we can now define, in order, the conditions that determine if the Country field is to appear:

- Zip Code > 9999 - enabled
- Town name = OTAWA - enabled
- Otherwise Zip Code < 10000 - hidden



Here we see that a record with a post code less than 10000 can still display the country field, if the town name is OTAWA.

Clicking the Designer button on the Forms Designer grid will trigger DesignBais to sort Sections and Subsections and to report if fields within any section overlap fields in other sections, based on the row position. The developer can then move fields, if required, such that the row positions of fields obey the rules required for proper section collapsing. Note that if a Subsection extends above or below the extent of its containing section then the Subsection row will define the extent of the Section.

Example of setup using subsections:

When there is something in RSK.SUMINS.LOR then

MainRIWP is displayed
MainRIWP_1 is displayed but disabled
MainRIWP_2 is displayed, enabled or disabled depending on RSK.PROP.Q2YN3

If there is a row overlap caused by a form element row span extending into the section below then collapsing will be disturbed. You need another rule to control the display as RSK.PROP.Q2YN3 comes into play after RSK.SUMMINS.LOR:

MainRIWP_1	No Change	RSK.SUMINS.LOR	# (Not Equal)		
MainRIWP_2	No Change	RSK.SUMINS.LOR	= (Equal To)		
MainRIWP_2	No Change	RSK.PROP.Q2YN3	# (Not Equal)	Y	
MainRIWP_2	No Change	RSK.PROP.Q2YN3	= (Equal To)	Y	

Example to demonstrate that subsections act like a case statement:
You may think that when DBWORK<BKT.TMD.DISP.WK> = null that this should work.

Clientdetails_3	No Change	BKT.DISP.CLIENT.WK	= (Equal To)	1
Clientdetails_3	No Change	BKT.DISP.CLIENT.WK	# (Not Equal)	1
Clientdetails_3	No Change	BKT.TMD.DISP.WK	# (Not Equal)	1

In fact the third rule will never be run. The section rules as shown above equate to this CASE structure:

```
BEGIN CASE
        CASE BKT.DISP.CLIENT.WK = 1 ;          Display
        CASE BKT.DISP.CLIENT.WK # 1 ;          Hide
        CASE BKT.TMD.DISP.WK # 1    ;          Won't happen due to above 2 cases
END CASE
```

Tab Index (Menu Option)

This menu option is used to define the input order of fields on a form. This option is used in conjunction with the 'Input Fields Use Tab Index' check box, located on the main Forms Designer form.

Prompts

Field Name List

Displays the list of all input fields and buttons that reside on the form. This list cannot be changed. Fields are initially sorted by row and column sequence and Tab Index is set in this sequence.

Tab Index

A number between 1 and 995 that represents the order the input fields are traversed when the user presses the Tab key. When the 'Input Fields Use Tab Index' check box on the main Forms Designer form is toggled then DesignBais resets the Tab Index values.

When Fields Use Tab Index is turned off all Tab Index values are set to zero. When turned on the Tab Index of input fields is assigned in row and column sequence with an increment of 5. Buttons are assigned a value of 995.

Within the Tab Index Properties form, changing any tab index will set the Input Fields Use Tab Index field on if it is not on.

Setting the Input Fields Use Tab Index check box off will set all tab index values to zero.
Warning: All bespoke tab sequencing that does not follow row and column sort sequence will be lost.

In the above example, the buttons are all assigned a tab-index of 995. This indicates that they will be traversed in the order that they appear on the form.

The placement of fields in a multivalue grid can be physically changed by changing their tab index, or by changing their column number:

- If Input Fields Use Tab Index is **on** then use Tab Index.
- If Input Fields Use Tab Index is **off** then use Column Number.

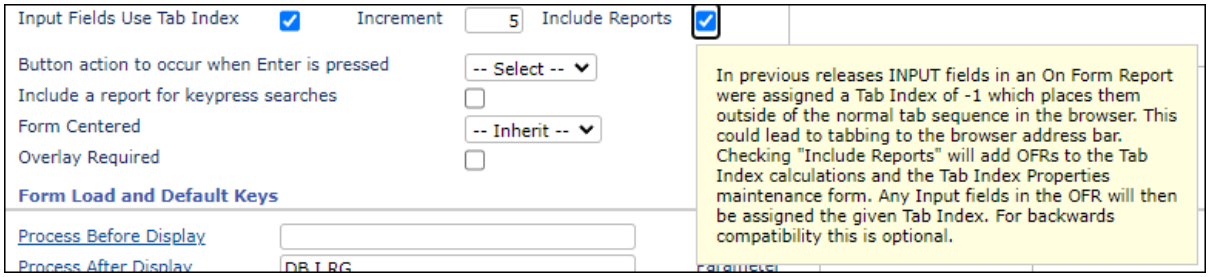
This is most easily done by using the Field List form within Forms Designer as this displays simultaneously the column number and tab index values for all fields in a grid.

Each time the Tab Index button is clicked the Tab Index values are reset to retain an increment gap of 5 between adjacent tab index values.

Note that tab index values for a particular multivalue grid must be contiguous. If a field that is not in the grid has a tab index value that lies within the range of tab indices of the fields that are in the grid then the tab sequence will not be followed after a change event. If a field in the grid, other than the last column of the grid, is amended then focus will not flow to the next field in that row, but will jump to the next row of the grid.

The limit of 995 is imposed by the browser. Browsers allow a maximum of 32000 (a bit more) tab indexes. Because DesignBais allows for 30 form layers the limit for each layer is about 1000. The value of 995 allows for some flexibility in the browser side processing.

In order to allow tabbing through on-form report input fields it is necessary to check the *Include Reports* option on the front Forms Designer screen.



Buttons

Submit

Commit the changes to the form and (if a Multivalue grid was re-organized) redisplay the designer form based on the new tab-index settings. Fields in the Tab Index are initially sorted by the Tab Index order. To see the new ordering, submit and re-open the Tab Index form.

Delete Field (Menu Option)

This menu option is used to delete the selected control/field on the designer form. The currently selected field will be deleted. (See the Undo Last Action menu option)

Insert Row (Menu Option)

This menu option is used to insert a row (one grid cell in depth = 10 pixels) to the form, thus shifting all objects from that point downward. You must first select the Insert Row menu option and then click on the grid location where you want the row inserted. This function is repeatable until another menu option is selected.

Delete Row (Menu Option)

This menu option is used to delete a row (one grid cell in depth = 10 pixels) from the form, thus shifting all objects from that point upward. You must first select the Delete Row menu option and then click on the grid location where you want the row deleted. This function is repeatable until another menu option is selected.

Insert Column (Menu Option)

This menu option is used to insert a column (one grid cell in with = 10 pixels) to the form, thus shifting all objects from that point to the right. You must first select the Insert Column menu option and then click on the grid location where you want the column inserted. This function is repeatable until another menu option is selected.

Delete Column (Menu Option)

This menu option is used to delete a column (one grid cell in width = 10 pixels) from the form, thus shifting all objects from that point to the left You must first select the Delete Column menu option and then click on the grid location where you want the column deleted. This function is repeatable until another menu option is selected.

Field List (Menu Option)

Displays a form showing all fields on the form together with a number of properties for each field.

1. Field Property
2. Field Name
3. Attribute
4. MV Flag
5. Process After
6. Field Read Use Variable - DBRECORD, DBWORK, DBOTHER.RECORD(n)
7. Field Text
8. Column
9. Row
10. Column Span
11. Row Span
12. Field Section
13. Justification
14. Tab Index

Changes made to these properties will be updated when the Submit button is clicked.

Note that if a field name is changed in this grid then DesignBais checks whether the field is used in section control. If so then a dialog like the one shown on the right allows the developer to confirm that the section control *Condition by Field* name will be updated to use the new field name.

Editor (Menu Option)

Opens the DesignBais Code Editor in a new adjacent tab. Note that a user must have the Code Editor Access field set to true in order to access the Code Editor. Refer to the User Maintenance and the Code Editor chapter.

Undo Last Action (Menu Option)

This menu option is used to undo the last action. The form will be reset as per the last action and any selected fields will be de-selected.

Up to release 8.5.1.6 there was no multiple undo feature. This required the developer to save “good” working versions of forms periodically. Cancelling a series of operations could only be performed by exiting the current form without saving and then reloading the form to continue from the last saved version.

In releases since 8.5.1.6 there are multiple undo steps available up until the Save or Exit button is clicked. The undo “images’ are stored on the DBIFORMS file with ids Filename*Formname.DESIGNER.SAVE (as the first available undo image) and then as Filename*Formname.DESIGNER.SAVEn where n commences at 1 and is incremented by 1 for each undo step. The .SAVE records that remain after a crash, or after a browser tab containing a designer session is closed via X, are cleaned out when the form is next read into Forms Designer and the Designer button is clicked.

If for some reason the development environment crashes (as a result of an event subroutine call failure) this option can be used to take you back to last action of the form. When this occurs, select the ‘Undo Last Action’ option as the very first action when entering back into the forms designer. This will recover any lost actions up to the last action.

Test On / Test Off (Menu Option)

This menu option is used to test the form that is currently being designed. All event types will be enabled. The form will function normally in this mode. All sub-forms can be called within this mode. Clicking Test On will hide all Forms Designer buttons leaving only the Test Off button. Click Test Off to turn test mode off and return to the form that is being developed.

If you are re-arranging read-groups it may be necessary to leave the forms designer and return. Read-groups are arranged when the form is originally loaded to improve run-time performance. As a result, any re-arrangement of read-groups may not be interpreted correctly until you leave and re-enter the forms designer. Similarly, some changes made to field properties may not be recognized until the form is reloaded. Save the current form, exit to the main Forms Designer page, then click Designer to come back in.

Rules (Menu Option)

This menu option accesses the Business Rules maintenance form.

Review (Menu Option)

This menu option opens the Form Fields form which allows you to review form fields, form reads, display class and derived fields and to highlight duplicate attributes.

The list of fields can be filtered to show only work fields, or only buttons etc.

The *Check Read Groups* option allows you to review the use of read groups across a set of forms.

The *Check Read Groups* option allows you to review the use of read groups across a set of forms.

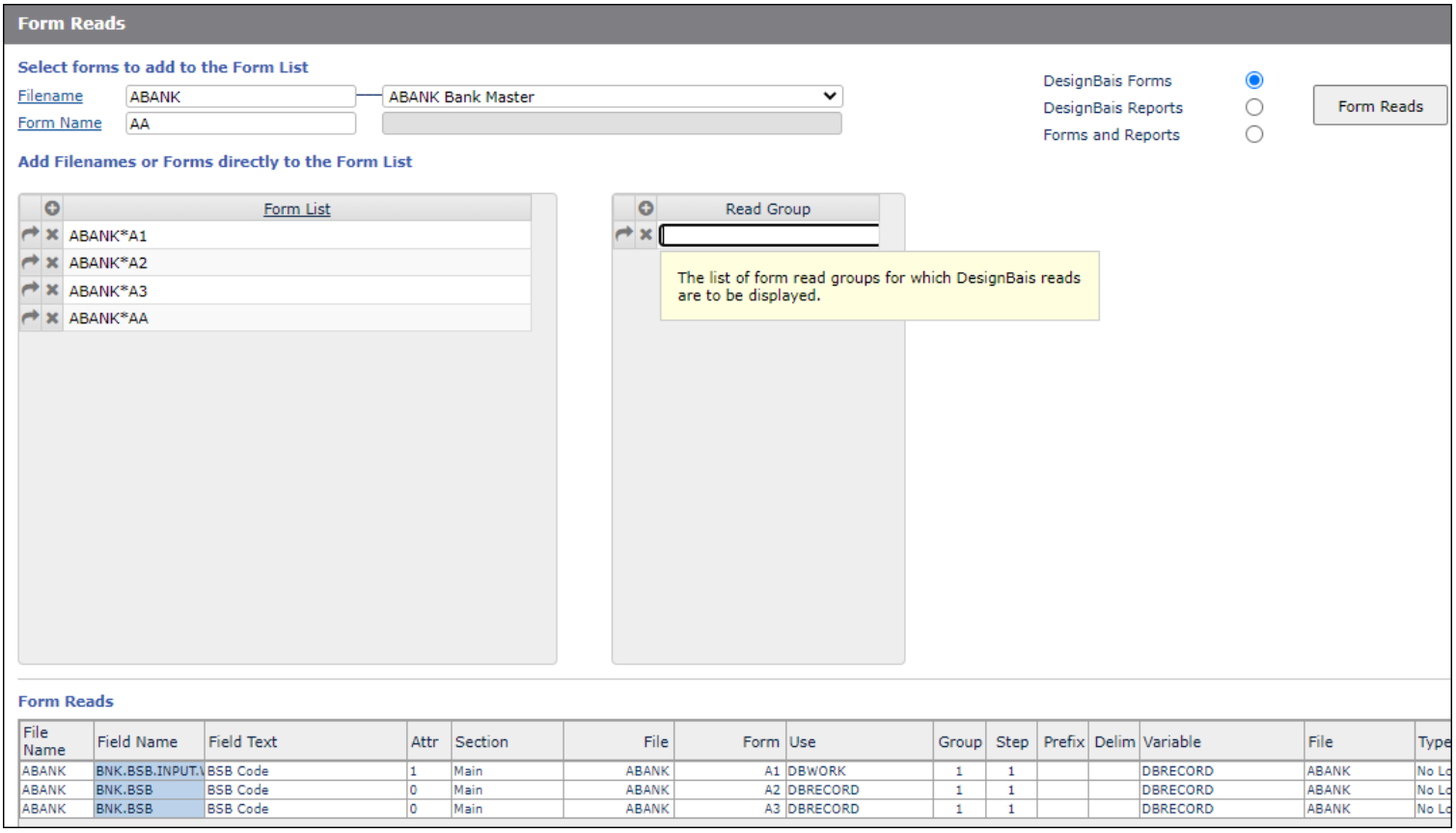
Note that in Release 7 / 8 a sub-form cannot re-use a read group or a read variable. This means that each form in a set of forms with the *Preserve Common* and *Sub Form* fields checked must use unique read groups and read variables. One form using read group 1 to populate DBRECORD and another sub form using another read group (such as read group 4) to populate DBRECORD will cause a problem. Similarly using read group 1 to populate DBRECORD and re-using read group 1 on another sub form to populate say DBOTHER.RECORD(1) will cause a problem.

Note that this restriction applies to reports and responsive design forms too.

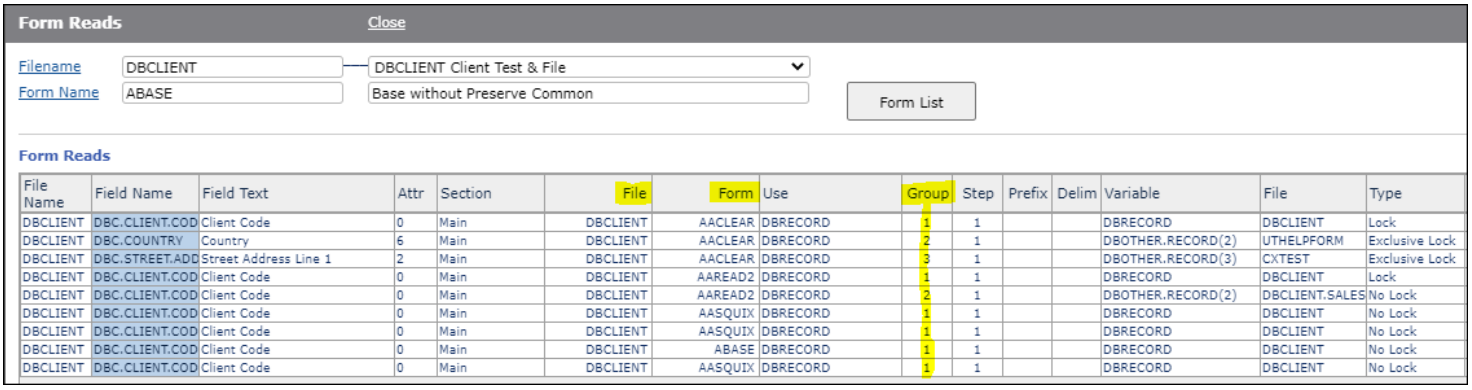
The *Check Read Groups* option allows you to enter a list of forms. Use the *Filename* and *Form Name* search buttons to select forms. As each form is selected it is added to the *Form List* grid below the search buttons.

You can also filter the Read Groups that are displayed by entering the groups to be displayed in the *Read Group* grid.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3



Then click the *Form Reads* button to display the read groups and read variables used in all the forms in the list. The developer can then verify whether there are clashing read groups or variables.



Multi Fields (Menu Option)

Use this menu option after selecting multiple field elements. Press Alt and click on one or more fields. Clicking Multi Fields opens the *Multiple Fields Property Update* form. This form permits the developer to move a group of fields, change the section name, change the display class, align, resize or to delete all of the selected fields.

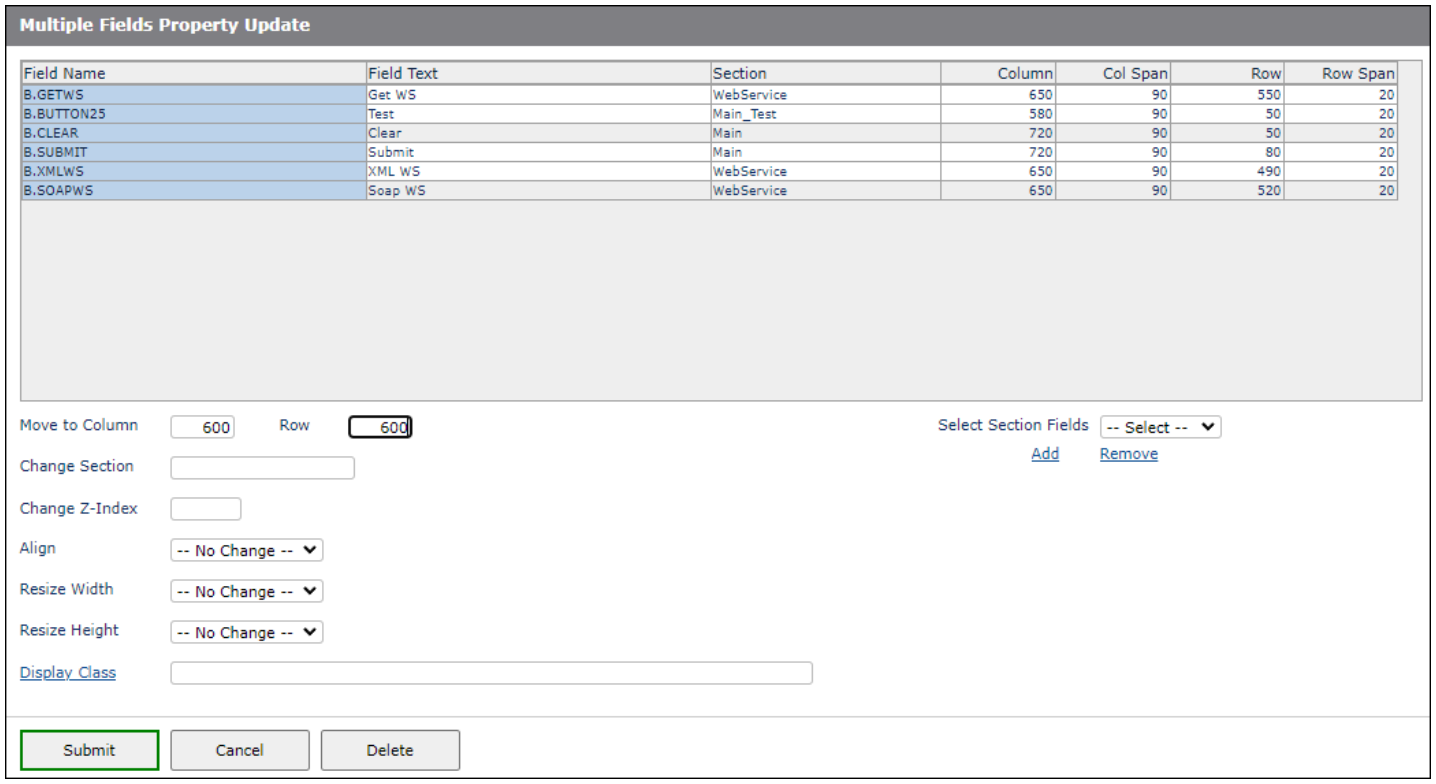
Fields are moved relative to the top left corner of the top most form control.

Fields may also be moved by clicking on the grid to place the top left corner of the top most control.

The Multi Fields form allows you to add or remove all fields in a section to or from the list of selected fields.

After clicking Submit the fields remain selected.

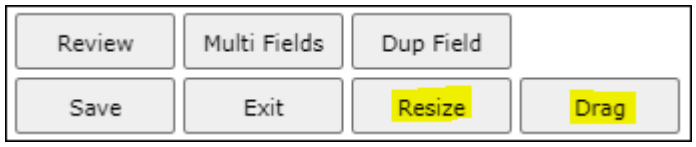
In the



Dup Field (Menu Option)

This will open the Add Field form with the values of the selected form. This saves re-entry of common properties e.g. a button style, the process after subroutine, etc. The properties may be amended before clicking the Add button to place the new control on the form.

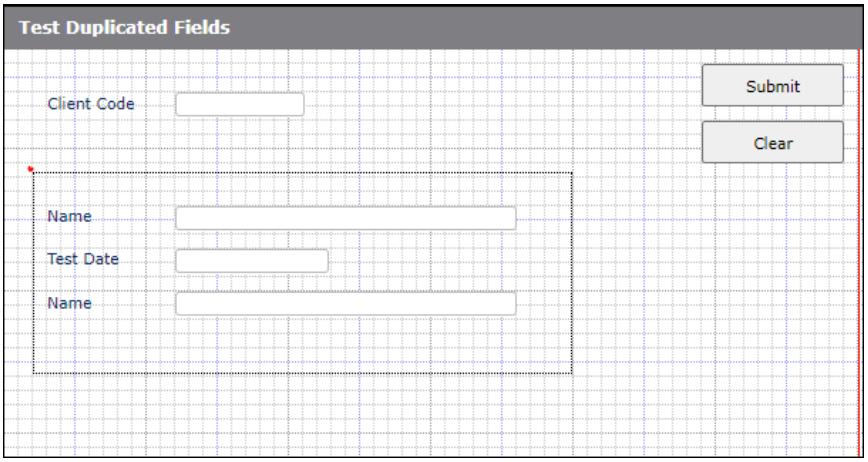
Lasso, Resize and Drag (Menu Options)



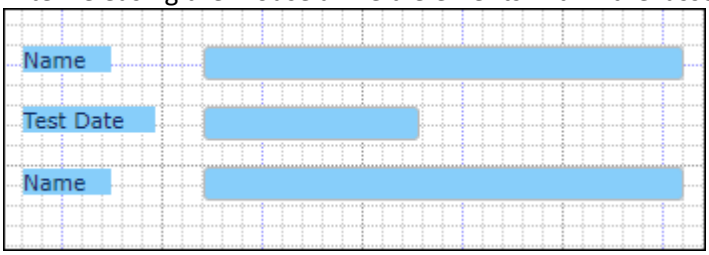
Buttons to enable field resizing and drag and drop using the mouse have been added.

Lasso

Forms Designer has a *Lasso* function. Click the canvas away from existing form elements then drag as shown here. The lasso will appear as a dotted line rectangle.



After releasing the mouse all field elements within the lasso will be highlighted.



Use *Alt Click* and drag at another point on the canvas to add additional elements to the *Lasso*.
This is equivalent to holding down the *Alt* key and selecting multiple fields.

Once a set of fields has been selected then the developer can move them all by clicking another point on the canvas, or delete them all using the *Delete Field* button, or update using the options available after clicking the *Multi Fields* button.

Use the *Undo* button to deselect all selected fields.

Resize (Menu Option)

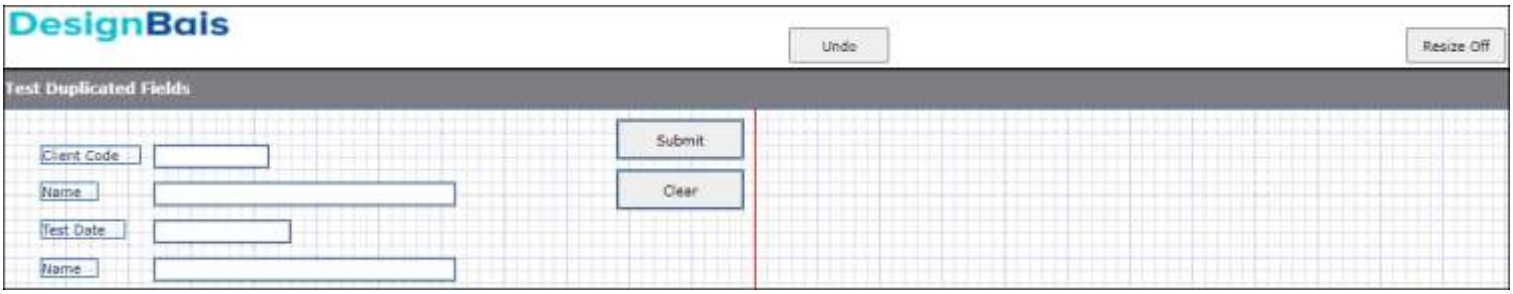
The *Resize* button sets field outlines that can be dragged to the right or downwards to change the column span or row span of fields. Note that setting resize outlines must wait for any CAPTCHA image to be displayed.

Resize Off removes the field outlines.

Fields may now be dragged into position. On some elements you can only drag via the border. On input elements the draggable box is normally larger than the element. The cursor will change when you can drag the element.

Resize example:

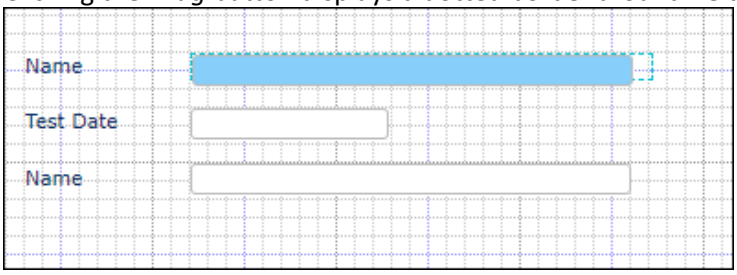
Clicking the *Resize* button displays the *Undo* button and the *Resize Off* button. A border appears around all fields that can be resized.



Place the mouse over the right hand edge or bottom edge of the field element that you wish to resize, click and drag. Release the mouse when the required column or row span is reached.

Drag (Menu Option)

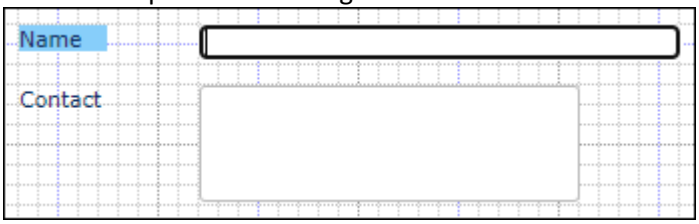
Clicking the *Drag* button displays a dotted border around field elements.



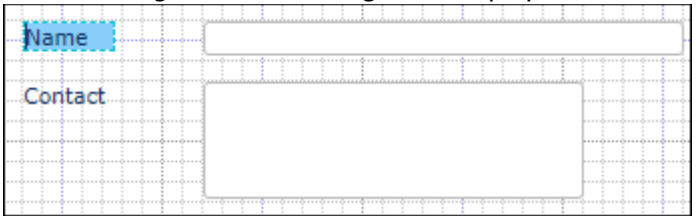
Click the field element then click within this border but outside of the field itself, then drag to field to a new position.

Multiple fields can be dragged by using the *Alt* key. Hold the *Alt* key and click a field in order to extend the drag net to include that field. Once the drag net extends past an unselected field it is not possible to then select that field.
It is best to use the *lasso* function to grad a set of field elements, before clicking the *Drag* button. Then when *Drag* is clicked all the fields within the lasso can be moved.
If you wish to extend the drag net using the *Alt* key then move across each row then down to the next row progressively.

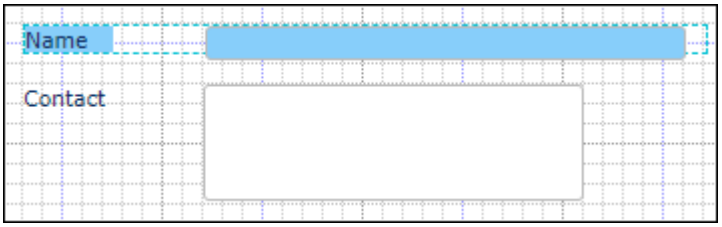
In this example the *Name* tag is clicked.



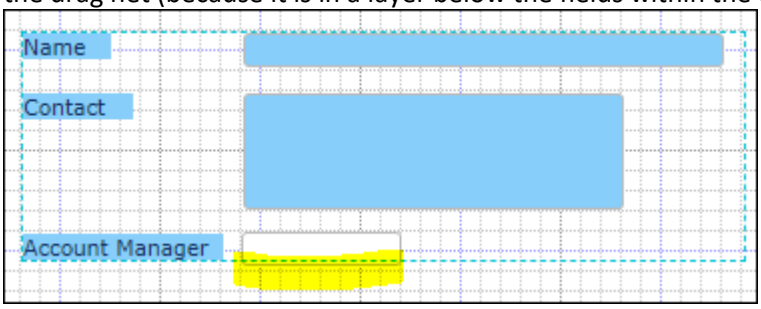
Click the *Drag* button. The drag net is displayed around the selected field.



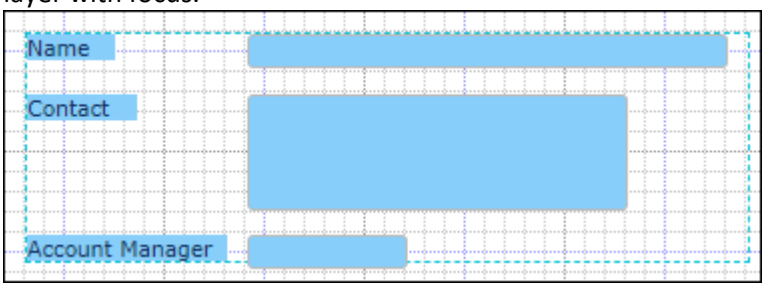
Hold the *Alt* key and click the name input field. The drag net extends around both selected fields.



Using the *Alt* key additional fields can be included, first the *Contact* tag, then the *Contact* input field, then the *Account Manager* tag. At this point however the *Account Manager* input field cannot be selected by clicking the portion within the drag net (because it is in a layer below the fields within the drag net).

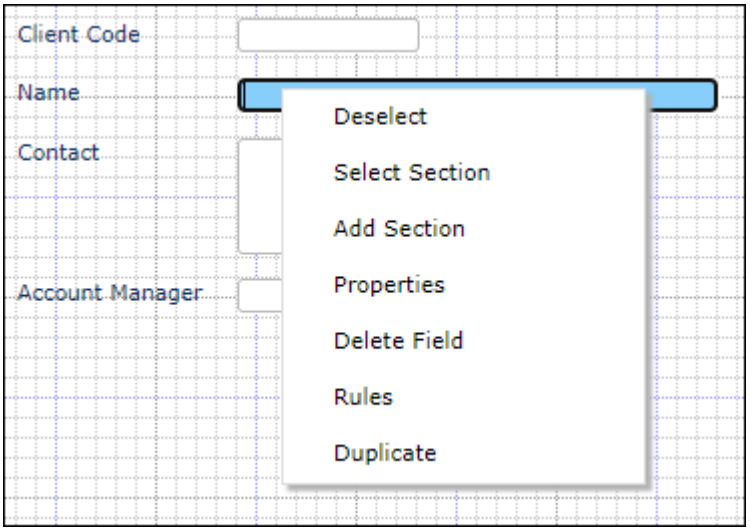


You can however click just below the dotted drag net line (highlighted in yellow) since this part of the field is still in the layer with focus.



Context Sensitive Menu

Right-click a form element to display the context sensitive menu.



Save (Menu Option)

This will save the form. It is prudent to save the form as you go. See notes under Undo Last Action.

Exit (Menu Option)

This will exit the forms designer and return to the main Forms Designer form. A prompt displays if the latest changes have not yet been saved.

Form Field Value Not Displayed

What to look for when a value in a field is not displaying on a form.

If the field is a DBWORK variable:

- Make sure that the field in Field Properties has the *Work Variable* checkbox set.
- Make sure that the field is linked to DBWORK in Field Properties in Forms Designer rather than, say, DBRECORD.

If the *Work Variable* field is accidently not checked when the field property is created, and the field is placed on a form that uses both *DBRECORD* and *DBWORK,* then Forms Designer may default the field to *DBRECORD*. If this is not corrected then a value placed in the field, by your basic subroutine for example, will not display.

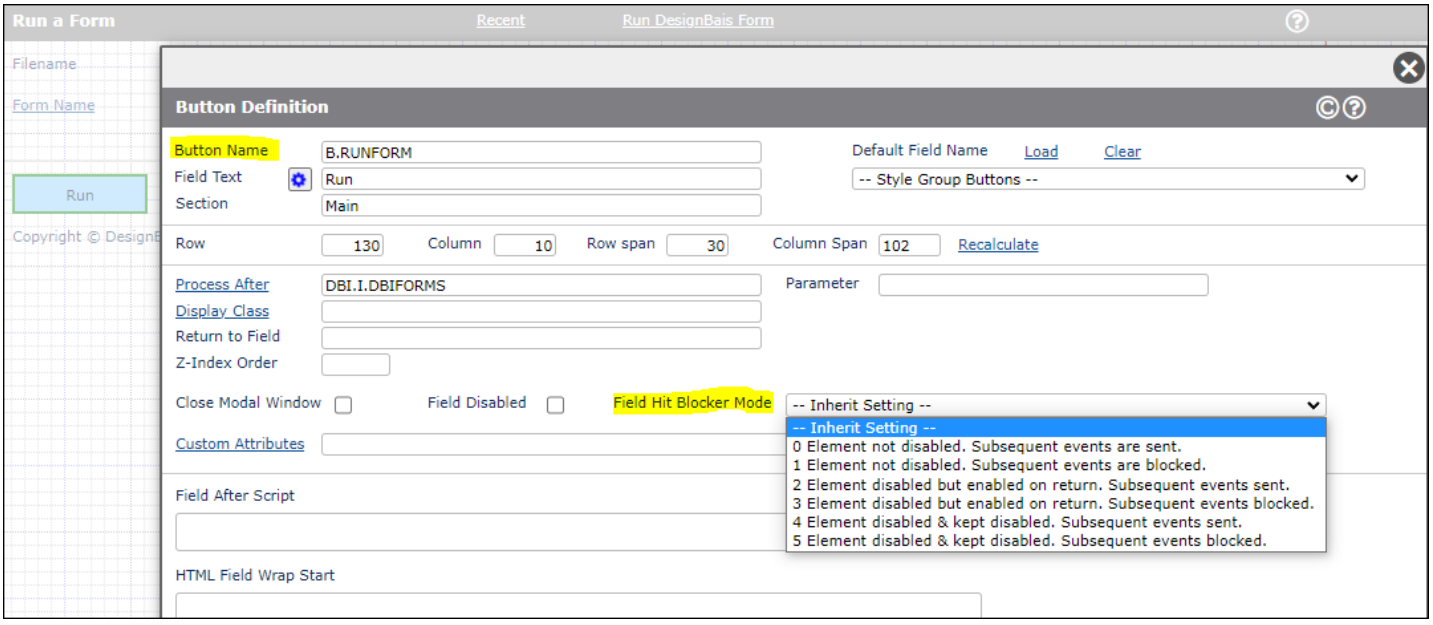
Make sure that all duplicate fields on a form have a subroutine call in the Process After slot, even if the call merely causes a server hit. The server hit allows DesignBais to set and refresh all field values in the browser.

Loss of an event such as a Button Click event

Check your Hit Blocker settings. Hit Blocker settings exist in Global Login Parameters and in System Parameters. The System Parameters setting can be set to *Inherit Global Setting*.

Hit Blocker action can then be set for Input fields and buttons on forms.

If the system setting is for example *Block Subsequent Events* and a particular button exhibits the loss of the button click event then it may be necessary to set the *Field Hit Blocker Mode* to *No Blocking.*



Chapter 9 – Report Designer

Report Designer

The Report Designer:

- is a graphical tool to create browser-based report forms.
- can create columnar style reports or pro-forma documents like invoices, statements and standard letters.
- renders reports in HTML which can be previewed on the desktop or printed directly to a printer.

The Report Designer is very similar in operation to the Forms Designer but differs in that a report form has multiple sections of **Header**, **Column Header**, **Detail**, and **Footer**. When designing a report the developer has a choice as to which sections to include.

A columnar report, for example, may have all sections included whereas a pro-forma letter may not have a Column header section.

Every report created will create two forms:

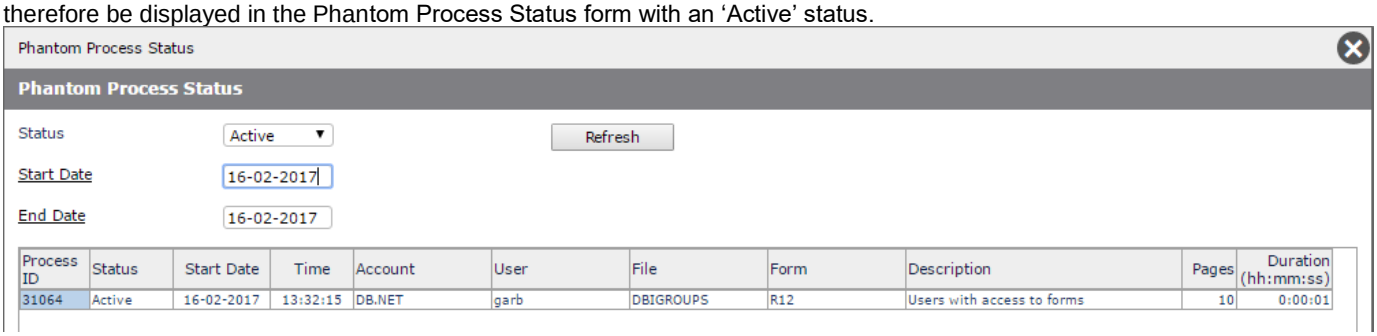
- The first has the same name as that of the report.
- The second has this name prefixed with 'RUN.'. This second form will be displayed when the end-user invokes the report. It is used to enter criteria to support the production of a report and also to make choices as to final output devices. The width of the RUN. form will be set to 600 px but is adjusted if user selection fields are present. Based on the field widths used in the user selection fields, which may contain a dropdown list, the form width may be extended up to a maximum width of 1000 px.

All reports are produced via a phantom process. This avoids any web-server invoked time-outs.

When developing reports each test run, and in Production, each run of the report, will create an entry in the &PH& file in Universe (_PH_ in UniData) recording the results of the selection for the report and any other output generated by the basic code used in the report. It is therefore possible to use the Basic CRT command to display variables as an aid to debugging the report.

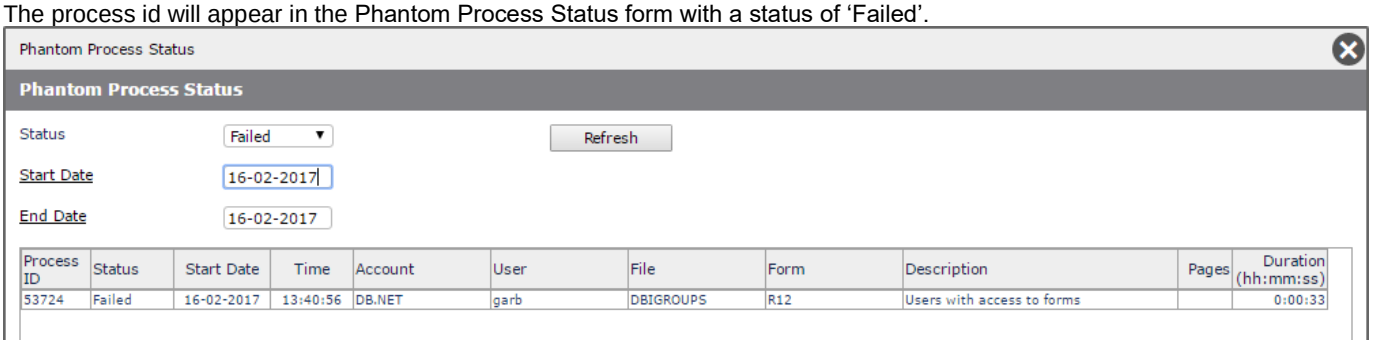
Monitoring Report Phantom Processes

DesignBais provides a method of monitoring report phantoms using the 'Phantom Status' button on the Active Users form (Form DBIPARMS_D20). If a report phantom has started and is still running then the Process Id will appear in the LISTU display and will therefore be displayed in the Phantom Process Status form with an 'Active' status.



The duration of the process will indicate potentially runaway or looping processes. These can be killed by clicking on the highlighted Process Id in the first column.If a report has been started and the phantom process is no longer in the LISTU display then this may indicate that the report has abnormally terminated.

The process id will appear in the Phantom Process Status form with a status of 'Failed'.



The records for failed processes remain until purged. Purging is controlled by the System Parameters Phantom Log Days setting.

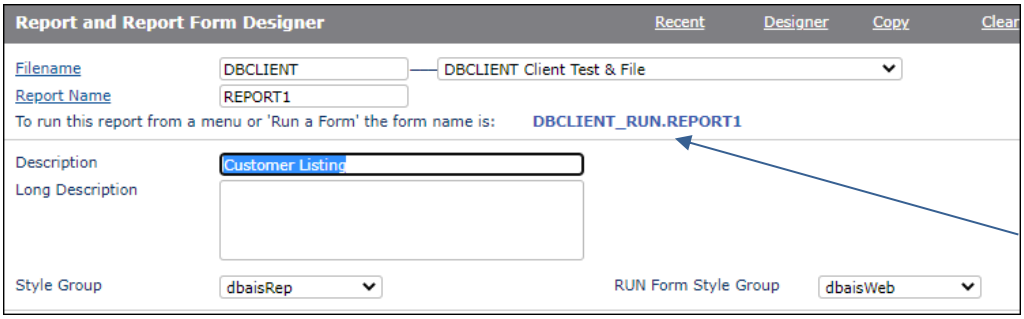
Refer to the section **Phantom Processing** in this manual.

If a report displays correctly when sent to a print page but not when it is converted to a PDF file then refer to the PDF Converter Font Percentage setting in the **General Global Parameters** section of this manual. This setting provides for a scaling factor to be applied to the report when it is converted to PDF format.

Report Designer Main Form

There are a number of sections to the report designer main form. In this document they have been separated for simplicity.

Section 1:

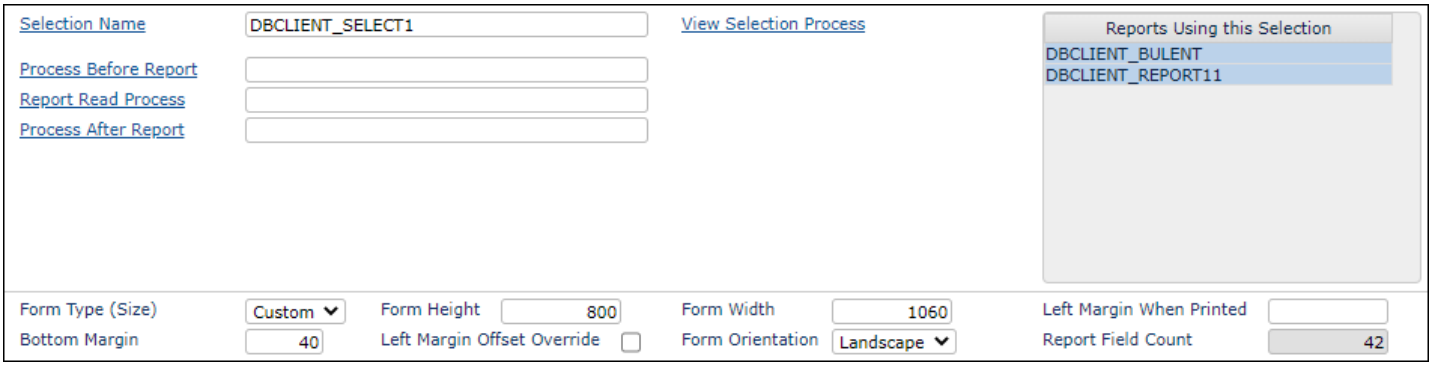


Name used to invoke the run report form.
Use this name to define the report in menus or when using the 'Run a Form' option

Prompts

Filename	Enter a name for a file that is known to DesignBais. There can be no underscores in the filename. You may also use the dropdown list to select a filename. Click on the Filename hyperlink to invoke a search for files that are defined to DesignBais.
Report Name	The name of the report to create or modify. Click on the Report Name hyperlink to invoke a search of reports for the selected file. The name can not include characters prohibited by DesignBais which are: _ (underscore), – (hyphen), * (asterisk), ~ (tilde) and | (pipe).
Description	Enter a description to support the report. This description will display on the report search form. This description will also display on the 'Run' form for the report.
Long Description	This is the full description of the report.
Style Group	The Style Group is used to set the standard look and feel properties of the report. DesignBais ships with a Style Group for reports called *dbaisRep*.
RUN Form Style Group	The Style Group to be applied to the RUN. form for the report. This allows RUN. forms to use the same style group as other forms in your application, allowing for a uniform look across the application.

Section 2:



Prompts

Selection Name — Assigns the Selection Process to be used for this report. The selection process must be defined before the report can be designed. The 'RUN' form will use the criteria questions gathered from the selection process and prompt the user for input at run time.

View Selection Process — Opens the Selection Process Definition form for the selection used by the report.

Reports Using this Selection — Displays the list of reports that use the selection process assigned to the current report. Each name is clickable and opens the report in Report Designer. On clicking submit or clear in Designer the current report is re-displayed.

Process Before Report — Provides for the inclusion of a subroutine call before the report is produced. This allows the developer to perform extra processing.

The process event for this process is "BEFORE REPORT".
Please refer to the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information

Report Read Process — Provides for the inclusion of a subroutine call after every record read in the report generation process. The process event for this process is "REPORT READ".

The construct of the report generation process is as follows:

```
Loop
        Read the next record from the selection list provided by the selection process
While there are records continue
        DBRECORD is read into read-group 1. (Reports always use DBRECORD and read-
        group 1).
        If there is a Process After Read subroutine identified then call the subroutine
        If there have been no errors generated from the after read subroutine, (IERR.TEXT = "")
        then    include the record in the report, otherwise skip the record
Repeat the loop
```

Please refer to the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information.

Process After Report — The name of a subroutine to call after the report is produced. Typically this can be used to modify the TAB delimited xls file produced from the report. The process event for this process is "AFTER REPORT".

Form Type (Size) — Used to identify the size of each page of the report. This is currently limited to *A4*, *Letter*, *Legal* paper sizes or the *Custom* option can be selected.

Form Type (Custom) — Selecting the *Custom* option will enable the Form Height, Form Width and Left Margin When Printed Prompts.

Form Height	Will set a custom height for the report.
Form Width	Will set a custom width for the report
Left Margin When Printed	By default, DesignBais will set a left Margin of about 40 Pixels (1 centimetre or 3/8 Inch). If a number is entered in this prompt, the left margin width will be set to this new pixel value.
Bottom Margin	The default bottom margin is 40 Pixels (about 10mm or 7/16 inch). When delivering reports via W3C mode it may be necessary to increase the bottom margin to allow for printer margin variations. Changes to the bottom margin affect the size of the form. It is recommended that the bottom margin should not be less than the default.
Left Margin Offset Override	DesignBais reports (by default) include a left margin offset of about 1/6 inch (4mm). In some instances you may wish to remove this. This can be achieved by checking this field.
Form Orientation	Used to identify the form (page) orientation for the report. The available options here are Portrait and Landscape. DesignBais cannot control the page orientation with IE11 or Edge and cannot detect the page orientation of any browser. DesignBais can control the page orientation in Chrome, Firefox and Safari. DesignBais sets the page orientation using: @page {size: A4 landscape; } Margins etc. are added to this. IE and Edge will ignore the "landscape" directive. Edge users will have to set page orientation using the printer settings.
Report Field Count	The number of fields currently on the report.

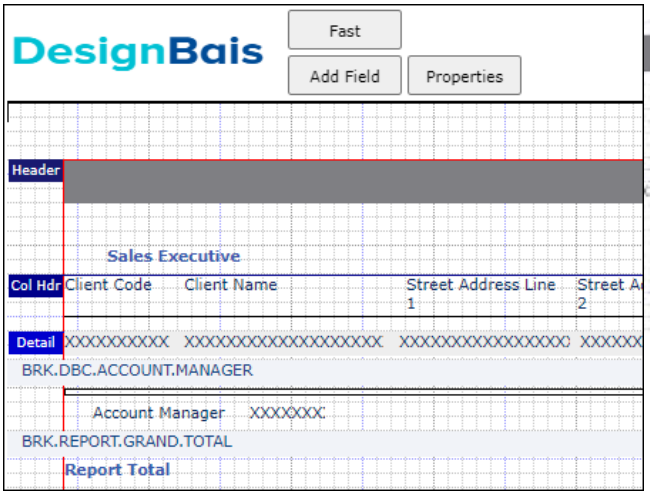
Section 3:

Include Header Section ☑
Include Column Header Section ☑
Include Detail Section ☑
Include Footer Section ☑

Prompts

Include Header Section

This check box indicates that the report will include a section for a header. The header is reproduced on every page of the report. If the Header Section is added to an already saved report then it may be hidden beneath the *Col Hdr* tag on the grid. Click on the *Col Hdr* tag and move it in order to reveal the *Header* tag.



Include Column Header Section

This check box indicates that the report will include column headers. Columns headers are located directly under the Header Section (if used) and are only added once per page. These are ideal for columnar-based reporting.

You may change the depth of this section by clicking on the 'Col Hdr' label and clicking on the canvas at the desired row.

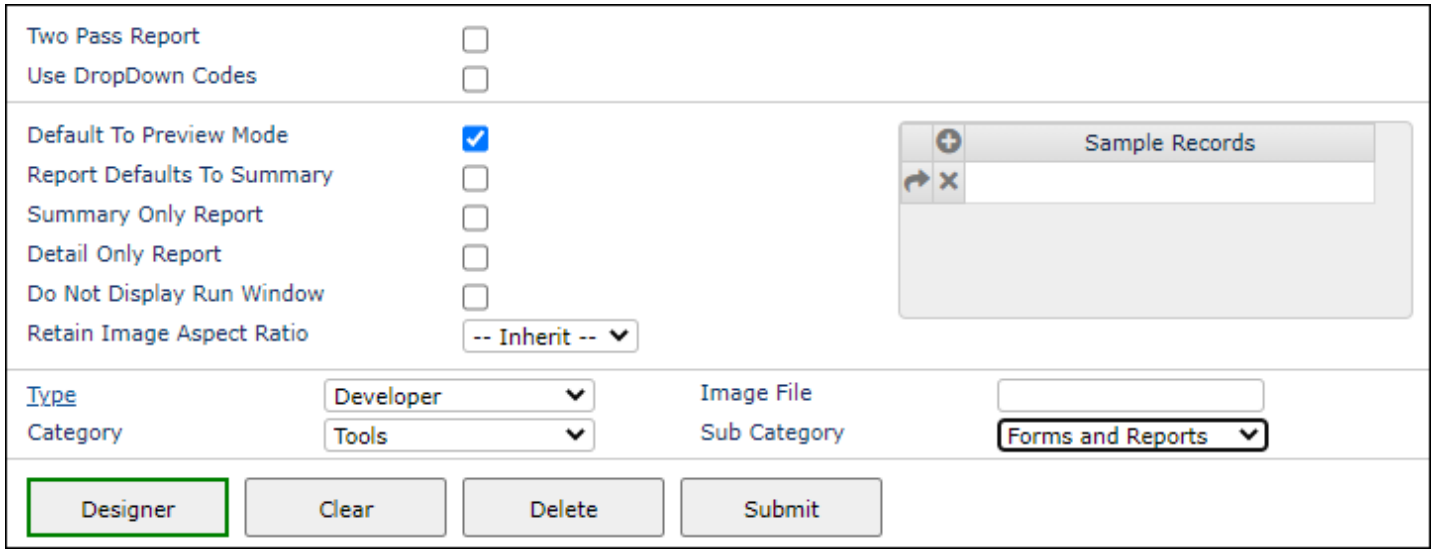
Include Detail Section

Will include a detail section in the report. The detail section is where the actual record will be placed. The detail section may be one or many lines in depth.

Include Footer Section

Will include a footer section in the report. This will print on the bottom of every page produced during the report production.

Section 4:



Prompts

Two Pass Report

This option will turn on two pass reporting. In a two pass report, DesignBais effectively runs the reports twice. The first pass of the report is to collect totals from each break section.

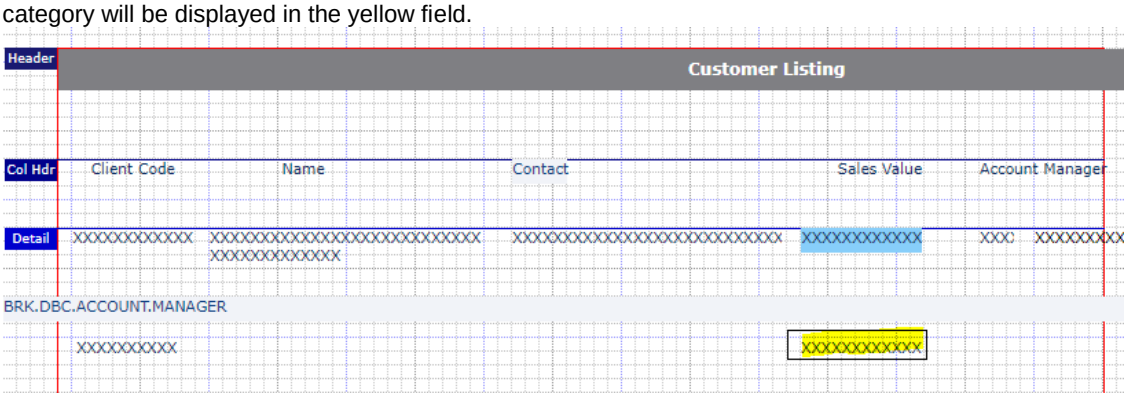
On the second pass, these totals can be referred to in the “REPORT READ” process.

Adding Break Fields and Totals

Every name of an accumulated field on a report is referred to in a field named DBACCUMULATORS.LIST. Totals for the current break levels are stored in the variable named DBBREAKTOTALS. This variable is attribute-delimited for each break level on the report. Each attribute is value delimited for each of the accumulated totals.

An example follows on the next page.

Note that break totals for a field are generated simply by repeating the field from the detail section in the break section. In the figure below the total for the Sales Value (blue field) for all detail lines within the break category will be displayed in the yellow field.



Client Code	Name	Month Sales	Year Sales
C00001	Max Monk	100.00	500.00
C00002	Harry Monk	100.00	400.00
Sales Executive	**Barry Miner**	**200.00**	**900.00**
C00003	George Basin	100.00	400.00
C00004	Margaret River	100.00	400.00
Sales Executive	**Sally Myers**	**200.00**	**800.00**
Report Total		**400.00**	**1700.00**

At all points in the second pass,

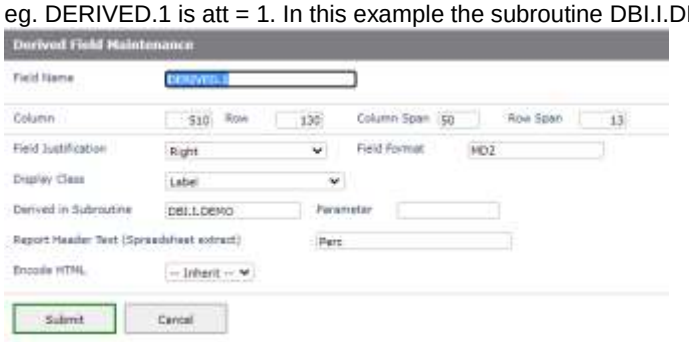
DBACCUMULATORS.LIST =]]MTD.SALES]YTD.SALES

(']' being @VM)

This enables the developer to locate the required total in the DBBREAKTOTALS variable.

Example of Usage:

A derived field must have a subroutine call that sets DBDERIVED<att> where the att is the the field suffix eg. DERIVED.1 is att = 1. In this example the subroutine DBI.I.DEMO will calculate the derived field value.



DBDERIVED can be set in the 'REPORT READ' event but the 'DERIVED' event will happen from the subroutine as shown above.

```
CASE EVENTSOURCE = 'DERIVED.1' AND SCREEN.NO = 'SCREEN.NAME'

    LOCATE "MTD.SALES" IN DBACCUMULATORS.LIST<1> SETTING POS THEN
        EXEC.MTD = DBBREAKTOTALS<1,POS> ;* Value associated with the Sales Executive
Break
        REP.TOT    = DBBREAKTOTALS<2,POS> ;* Value associated with the Report Total.
        DERIVED.1 = (DBRECORD<MTD.SALES> / EXEC.MTD) * 100  ; * First % Column
        DERIVED.2 = (DBRECORD<MTD.SALES> / REP.TOT) * 100 ; * Second % Column
END ELSE
        DERIVED.1 = 0
        DERIVED.2 = 0
END
```

At this point we now have the Executive total and the Report total. It is then possible to establish what the percentage of each total the client "C00001" [line 1] contributes to both the Executive total and the Report Total. These amounts could then be added to the report as follows:

Client Code	Name	Month Sales	%	Year Sales	%
C00001	Max Monk	100.00	50	500.00	55
C00002	Harry Monk	100.00	50	400.00	45
Sales Executive	**Barry Miner**	**200.00**	**50**	**900.00**	**53**

C00003	George Basin	100.00	50	400.00	50
C00004	Margaret River	100.00	50	400.00	50
Sales Executive	**Sally Myers**	**200.00**	**50**	**800.00**	**47**
Report Total		**400.00**	**100**	**1700.00**	**100**

Break % totals are accumulated during the subroutine call in break fields.

As noted above totals are accumulated based only on *Field Name* without regard to the *Variable to Use* that applies to the field. If the same field name is used in the the detail section (with *Variable to Use* of DBRECORD) and in the break total or grand total section (with *Variable to Use* of DBOTHER.RECORD(n)) then totals for this field, if it has a type of *numeric*, will appear as a single total for both single value and multi-value fields. If the type of field is *alpha* then multi-value fields will be presented in the total as distinct values on multiple lines.

In order to display distinct values on multiple lines for *numeric* fields ensure that the field name used in the break total and/or grand total section differs from the field name in the detail section, even though the fields may reference the same attribute number.
The following code snippet provides a very simple example to demonstrate how totals can be accumulated within the subroutine referenced by the DesignBais report. This example is from the REPORT READ event. Here the data is extracted from DBRECORD and accumulated in a multi-valued attribute held in DBOTHER.RECORD(2).

```
JMAX = DCOUNT(DBRECORD<202>,VM)
FOR J=1 TO JMAX
   CDE = DBRECORD<202,J>
   AMT = DBRECORD<203,J>
   LOCATE CDE IN DBOTHER.RECORD(2)<202>,1 SETTING POS ELSE
      INS CDE BEFORE DBOTHER.RECORD(2)<202,POS>
      INS 0 BEFORE DBOTHER.RECORD(2)<203,POS>
   END
   DBOTHER.RECORD(2)<203,POS> += AMT
NEXT J
```

In this example the same field name may be used to reference both the *Variable to Use* DBRECORD and DBOTHER.RECORD(2). In that case the grand total will appear as a single value. By changing the name of the field in the grand total section of the Report Definition, even though this field name still references attribute 203, the grand total will display as distinct lines, one line for each multi-value in attribute 203.

Note that DesignBais can only total multi-value numeric fields as it has no mechanism for controlling the correct display sequence, as shown in the example above, where there is an associated description or code field.

Use Dropdown Codes

By default DesignBais will include the descriptions assigned to field-based dropdowns on a report. To change this and include the codes instead, check this prompt. This will print the codes associated with field-based dropdowns.

Default to Preview Mode

This check box is used to determine whether Preview Mode is selected (by default) at runtime. If checked, the user will be able to preview the report before printing. The user can override this setting at runtime.

Report Defaults to Summary

Determines whether the report will default to Summary Mode. This indicates that the report will only produce break and grand totals and the detail section will not be printed by default.

Summary Only Report

Check this box if the report is to be run in Summary Only Mode

Detail Only Report

Check this box if the report is to be run in Detail Only Mode

Do Not Display Run Window

Check this box to suppress the report run window. This will produce the report to the users default printer in the default mode set by the 'Default to Preview Mode' prompt. You can change the style of printing by using the Parameter associated with the report process.

Eg. PROCESS.PARAMETER =

“REPORTPRINT”	Run the report in non-preview mode
“REPORTPREVIEW”	Run the report in Preview mode
“REPORTEMAIL”	Run the report to email

Retain Image Aspect Ratio

By default, images on the report are resized to fit 100% of the width if the container is portrait or to fit 100% of the height if the image container is landscape. The maximum width and height are also set to limit the image to its container. This can lead to distortion unless all images are sized the same.

Select *Yes* or *Inherit* from the "Retain Image Aspect Ratio" dropdown to allow images to be restricted to the container size but displayed without distortion. This will mean that images may not fill the report container.

This can be set here for all images on the report. Individual images may override this setting. *Inherit* has been included to pick up a System wide setting which is not yet implemented. If no System Parameter exists then the default will be *No* to leave images unchanged.

Sample Records

This Multivalue prompt provides for the entry of keys to use when testing the report. This is useful to speed the testing process, as the selection is by-passed. The sample records are only used in test mode.

Type
Image File
Category
Sub Category

These fields allow the user to define Type, Category and Sub Category for the report. These are used in filtering the display of Favorites (Favourites). Refer to these fields in the Forms Designer section above for more details.

Buttons

Designer — Will invoke the report designer

Clear — Will clear the form and not commit any changes made.

Delete — Will delete the report.

Submit — Will save the report definition on the DBIREPORTS file.

Report Designer Usage

The first thing you are presented with in the report designer is a canvas with a grid background. The grid has a size of ten by ten pixels. There is a blue line indicating the fifty pixel and a dark grey indicating the one hundred pixel positions.

The canvas includes a red square that is used to indicate the boundary of the report.

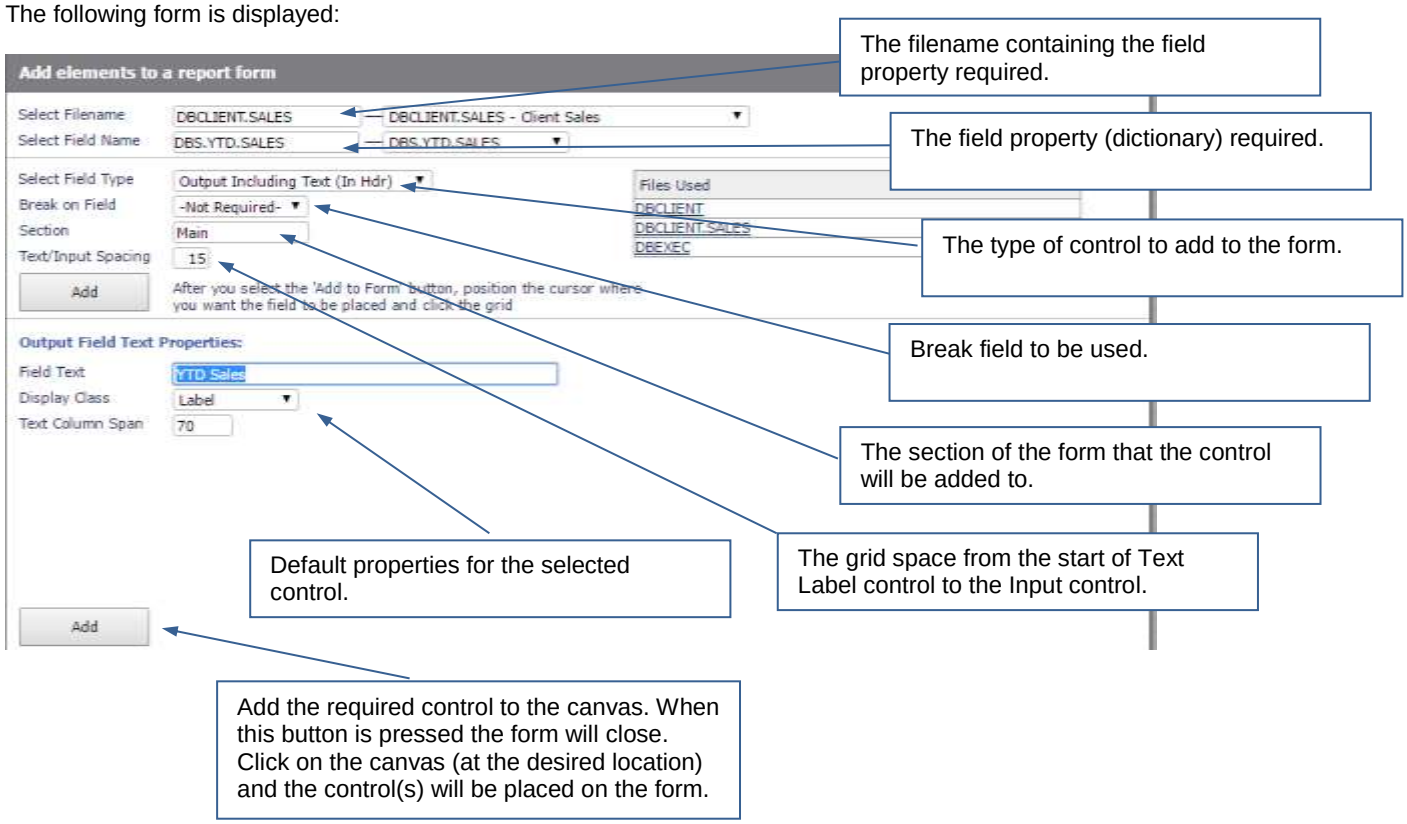
Any control placed outside of the boundary (you should always use the right side) will not be included in the report, but can be used to provide details for the report. An example is where you wish to provide a translated description on the report, but not it's supporting code. The supporting code field (which can include the read) can be place outside of the report boundary. The resulting description can be placed within the boundary.

From Version 7 the Report Designer uses class="style" instead of extracting the basic style elements from the style definition. Developers should note that certain style elements may affect the size of the created or indeed the size of the available space for the text (padding, borders, box-sizing, etc).

Add Field menu option

To add a control to the form, select the **Add Field** option from the side menu.

The following form is displayed:

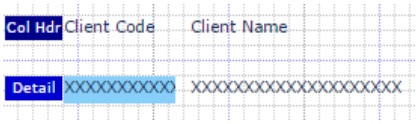


There is a **limit of 500** field elements for a report, or for a set of reports invoked using the DBREPORTLIST.

Field names must not begin with "BRK." since this is a reserved prefix within the DesignBais Report Designer. DesignBais introduces break fields with this prefix. If introduced by the developer then array subscript out of range errors may occur.

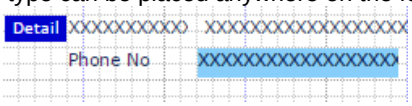
Output Including Text (In Hdr)

Places an output only field on the report form including the supporting text from the Field Property. This will place the supporting text from the output field in the Column Header section and the actual data element in the detail section.



Output Including Text (In Detail)

Places an output field on the form with the supporting text label from the Field property.
This field type can be placed anywhere on the form (not just the detail section).



Output Field Only

Places an output field on the report form. No supporting text label is loaded from the field property.

Text Field

Places the supporting text from a Field Property on the report form.

Text Only

Places a field on the form as a text label. This is commonly used for headings or supporting text on a form.

Text fields can include system variables to add detail to the report.

These are:

@DATE	Displays the system date on which the report was produced (month is upper case)
@LONGDATE	Displays the system date on which the report was produced in long date format (month in text case)
@TIME	Displays the system time as at the commencement of the report
@WEBLOGON	Displays the user logon code.
@DBIACCOUNT	Displays the current database account or directory name
@PAGE	Displays the current page number on every page printed
@PAGES	Displays the total number of pages on the report.
@REPORT	Displays the report name
@SECURITY	Displays any message from the DesignBais security system during report runtime.
@SELECT	Displays the selection statement used for the report.
@EMAILPRINT	Must be placed within the report header to allow pages to be directed to 1 or more email addresses.
@DBUSAGEVAR<attr,val,subval>	Displays the contents of DBUSAGEVAR. Eg DBUSAGEVAR<11,1,2> displays the 2nd sub value in value 1 of attribute 11.

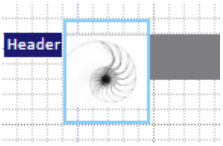
These system variables can be used together in the same field.

Example: Date @DATE - @TIME

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

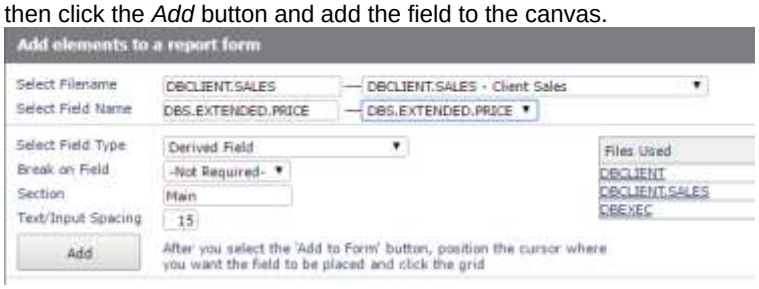
Image

Places an image control on the report form. The image can be any type that will display in a browser. The default image is the DesignBais logo. Use the Properties option to change the default image.



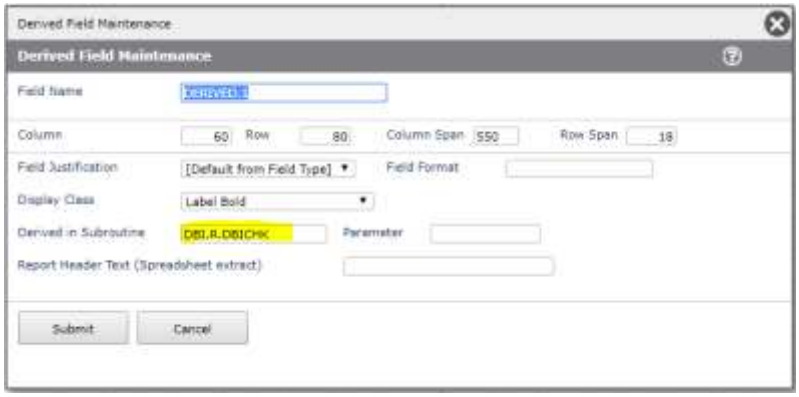
Derived Field

A Derived Field is a report field that does not pertain to a field on a file. It's a field created just for the report. Derived fields can have any name [the default is DERIVED*.n*. The only mandatory element of the name is the "**.**" and an attribute reference. As all derived fields are stored in the DesignBais common variable DBDERIVED, the attribute reference determines which attribute the value is stored in. When adding a Derived field ignore the 'Select Field Name' field which will display a field name from the 'Select Filename'. This field is not relevant to the Derived Field option. The procedure therefore is to click the *Add Field* button, select *Derived Field* from the *Select Field Type* dropdown, amend the section if required and then click the *Add* button and add the field to the canvas.



Then click the *Properties* button in Designer. The Field Name will default to *DERIVED.n* where *n* is the next available attribute in DBDERIVED. The name can be changed as long as the prefix *.n* is retained.

The derived field must be populated by either basic code in a subroutine or via a Code Block. To generate a Code Block enter *C:* in the *Derived in Subroutine* field. To use a subroutine the name of an existing catalogged routine must be entered. If the subroutine does not yet exist leave the *Derived in Subroutine* field blank, since the field validation prevents entry of a name that does not have a catalog entry. The name can be entered later.



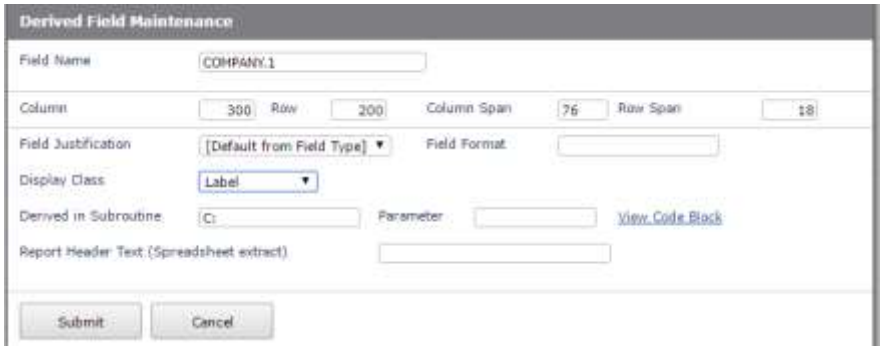
Example:

You have a Derived field named *ANYNAME.1* on your report.

In your Basic program update the value as *DBDERIVED<1> = value*.
Note that the update needs to be placed in the *REPORT READ* event.

Make sure that the subroutine name is defined in the *Report Read Process* slot on the front screen of the Report Designer.

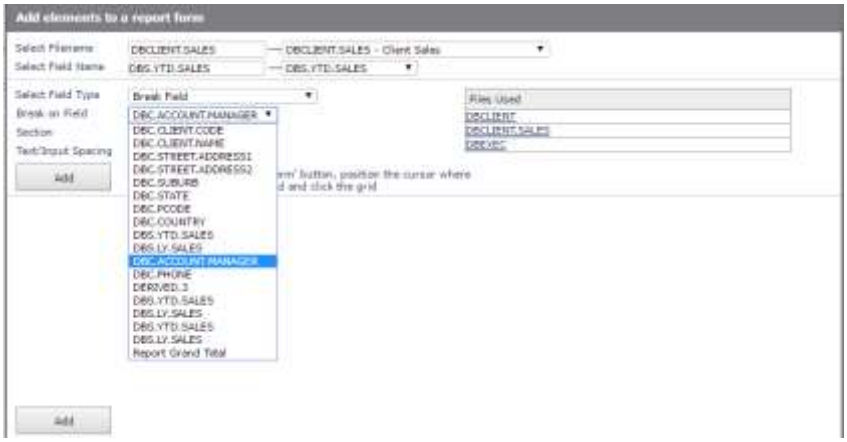
In a code block update the value as *ANYNAME.1 = value*.


Derived Field Maintenance
Field Name
COMPANY.1
Column
300
Row
200
Column Span
76
Row Span
18
Field Justification
[Default from Field Type]
Field Format
Display Class
Label
Derived in Subroutine
C:
Parameter
View Code Block
Report Header Text (Spreadsheet extract)
Submit
Cancel

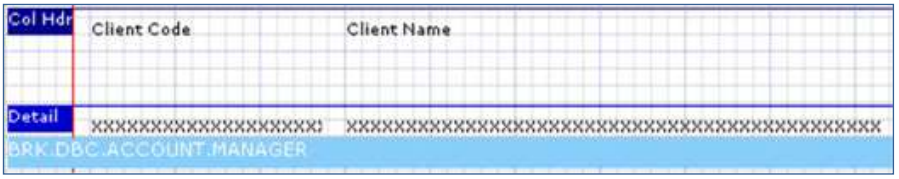
Break Field

Will add a break field on the report. The distance between the detail line and the first break field will determine the depth of the detail section. When adding a Break field ignore the 'Select Field Name' field which will display a field name from the 'Select Filename'. This field is not relevant to the Break Field option.

When selecting the 'Break Field' option, the 'Break on Field' prompt must be selected. All fields placed on the form are available to be used for a break field.

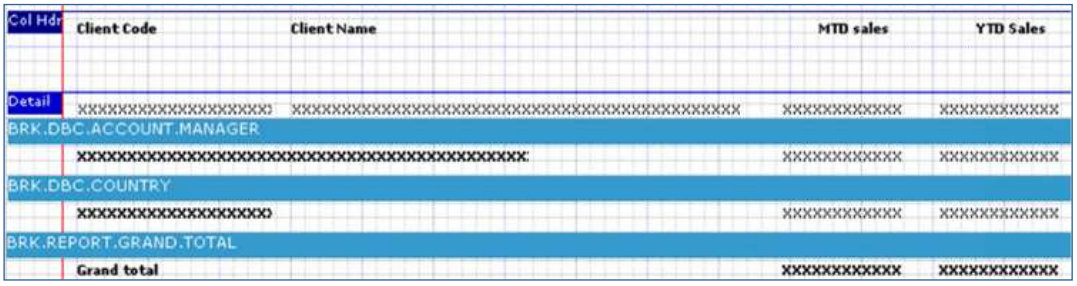


After the selection of the Break Field option, the 'Break on field' prompt contains all of the available break-on options.



In the above example, the break field DBC.ACCOUNT.MANAGER was added as a break field. The break field is named BRK.DBC.ACCOUNT.MANAGER. It is possible to set page-break and process after parameters for the break field by simply selecting properties from the side menu.

The break construct allows for fields to be placed on the form below the break line that will only be included on the report with the break-value changes.

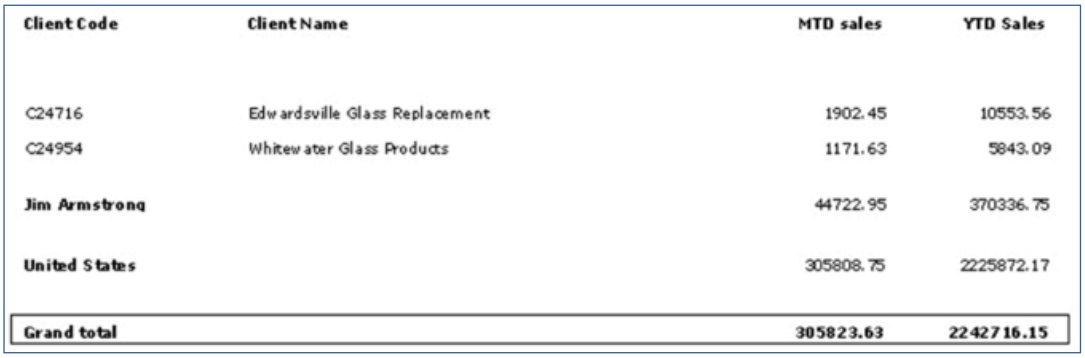


The results are as follows:

Client Code	Client Name	MTD sales	YTD Sales
C24716	Edwardsville Glass Replacement	1902.45	10553.56
C24954	Whitewater Glass Products	1171.63	5843.09
Jim Armstrong		44722.95	370336.75
United States		305808.75	2225872.17
Grand total		**305823.63**	**2242716.15**

Box/Line

Adds a box or a line to the report. A line is simply a box with a row span of 1.



Client Code	Client Name	MTD sales	YTD Sales
C24716	Edwardsville Glass Replacement	1902.45	10553.56
C24954	Whitewater Glass Products	1171.63	5843.09
Jim Armstrong		44722.95	370336.75
United States		305808.75	2225872.17
Grand total		**305823.63**	**2242716.15**

Report Designer Properties (Menu Option)

There are various properties associated with fields added to a report form. The following describes the available properties and what control the properties pertain to.

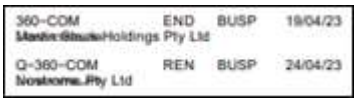
Properties are displayed when the menu option, **Properties** is selected from the Forms Designer top button row.

Properties:

Position, width and depth Properties

Row	All controls Controls the row (Pixel on the y-axis) that the control starts in.
Column	All controls Controls the column (Pixel on the x-axis) that the control starts in.
Row Span	All controls Controls the depth of a field in Pixels. Images scale width in relation to height set in Row Span.
Column Span	All controls except the Image control. This field controls the width of a field in Pixels.
Alternate Length	Output Fields Controls the length of input field

The row and column span of a report field may cause the field contents to overflow into an adjacent field as shown in the snip to the right. Here the text is wrapping at a white space character.



The developer can set the row span to make the field box deep enough to allow a second line of text. In this case the second line overlaps the field below.

To overcome this the developer can stop the field from overlapping the one below by making its row span shorter.

If the field still wraps then add “white-space:nowrap” to the style of the field. This stops the text from wrapping and will allow it to fill the allocated box horizontally. The text may still be truncated. If there is space in the report layout the devloper could make the field wider but again must avoid overlap with the next field.

Display Properties

Justification

Select the justification attributes for the current field

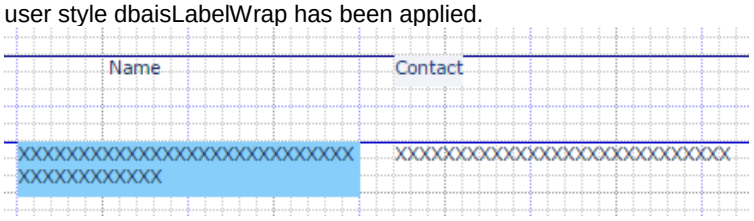
Field Format

By default, DesignBais will use formatting from the Field Properties. There may be instances where an output conversion is to be specified that is not the default.

Display Class

Text Labels and Output Fields

Sets the display characteristics for the field. The names used correspond with the style group settings. Note that from Version 7 onwards Display Class is case sensitive.
Version 7/8 also includes a new class dbaisLabelWrap which allows text fields to be wrapped over 2 or more lines. In the example below the row span of the Name field has been increased from 18 to 32 and the user style dbaisLabelWrap has been applied.



Suppress Repetition

If you wish to suppress repetition of output for the currently selected field then set the check box to true. This will stop data being repeated in consecutive rows on a report.

Encode HTML

(From Release 8.3.1.2) Output fields containing HTML elements should be encoded to prevent XSS injection attacks. Encoding HTML means converting to their display-only string those characters that have meaning in HTML.
For example the “less than” character “<” is encoded as & l t ; (without the spaces).

This means that any HTML will display as entered but will not be active in the browser.
Encoding may be set to Inherit, Yes or No.

Encoding may be set on Output Fields. If an Output Field is set to *Inherit* (the default action) then the System Parameter setting will be applied. If the System Parameter value is *Inherit* then the Global Setting will be applied. The Global Setting will default to No encoding to provide backwards compatibility for existing applications.

It is recommended that HTML Encoding be turned on globally. Only fields requiring active HTML elements should be set to *No encoding*.

Note that if *Encode HTML* is set to *No* and the value in the field contains HTML commands that increase the font size for example, then the rowspan of the report field may need to be increased. In the case of a multi-value field where normally a single row with rowspan 18 allows multiple rows to print based on the data, it is necessary to increase the rowspan of the field element to embrace the area in which the multiple values are to print.

Section Property

Section

All Controls

Default is ‘Main’. Determines the section that the control belongs to on a report form. A section can be displayed or hidden based on run-time properties associated with the controls on the report form.

Please see Sections later in this chapter.

Image – Specific Properties

Image File

Image Only

Defines the name of the image to use. The image must be loaded into the images directory which is a child to the DesignBais site physical directory. You may create a sub-directory under the existing images directory to store images e.g. Images\myimages. If this is the case the image name entered for this property would be myimages\imagename.

Multiple images can be loaded into a work field on a report. The following example will assist the developer.

Assume that the DesignBais report has a Report Field Definition linked to a work field DBWORK<RPT.MV5.WK>.

In the code sample below the style properties are added to the image path:

style="max-width:100%;max-height:100%;"

This makes the image fit the MV field width or height as defined in the report definition.

Note that wide images may appear small but the entire image will be visible. If the image dimensions are known then the style sheet css file can be used to rotate the image so that the longest measurement goes down the page. If printing in portrait then it is best to use images that have been rotated during capture.

```
DOC.IMAGES contains a multivalued list of images
PATH = “some path”
PATH.HTML.PROFILE = PATH:"/profile/"
ROOT.URL = FIELD(PARMS.REC<1>,'/',1,4)
*
PROF.IMG.PATH = '<span>%%%</span><br></br><img src="':ROOT.URL:'/':PATH.HTML.PROFILE
PATH.SFFX = '" style="max-width:100%;max-height:100%;" alt=" " ' : ' "?nc="':DATE():TIME():">'"
*
IMAX = DCOUNT(DOC.IMAGES, @VM)
FOR II = 1 TO IMAX
   IMG.ID = DOC.IMAGES<1,II>
   READ IMAGE.REC FROM F.IMAGE, IMG.ID ELSE CONTINUE
   OUTPUT.FILENAME = IMAGE.REC<IMG.ORIG.FILENAME,1>
   IMG.PATH = PROF.IMG.PATH : IMAGE.REC<IMG.FILENAME,1> : PATH.SFFX
   IMG.PATH = CHANGE(IMG.PATH,'%%%',OUTPUT.FILENAME,1)
   IMG.CTR += 1
   IMGS.PATHS<1, IMG.CTR> = IMG.PATH
NEXT IMG.NO
DBWORK<RPT.MV5.WK> = IMGS.PATHS
```

Box/Line – Specific Properties

Line Color

Defines the color of a box/line. Colors can be entered as either, one of the 140 Color names assigned for a browser, or as Red, Green, Blue (RGB) hex values entered as #RRGGBB, eg. #EEEEFF. The '#' is mandatory for all RGB values entered.

Any internet search engine (Google, Yahoo etc), will return a number of sites that contain a full list of the 140 color names. Simply search for "140 colors".

Background Color

Defines the background color of a box/line. Colors can be entered as either, one of the 140 Color names assigned for a browser, or as Red, Green, Blue (RGB) hex values entered as #RRGGBB, eg. #EEEEFF. The '#' is mandatory for all RBG values entered.

Controlling Reads on Reports

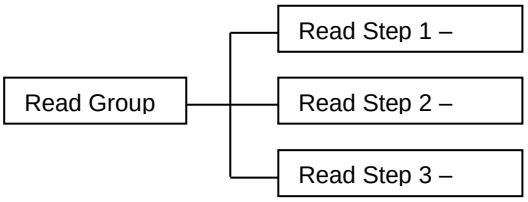
Read Group

Output Fields

The Read Group is used in a report to define additional reads. DBRECORD is automatically assigned to read group 1 in a report. If you re-use read group 1 or DBRECORD (which is possible) on an output field then this will destroy the contents of DBRECORD read as part of the reporting process.

The read-group structure may be set-up as follows:

Eg.



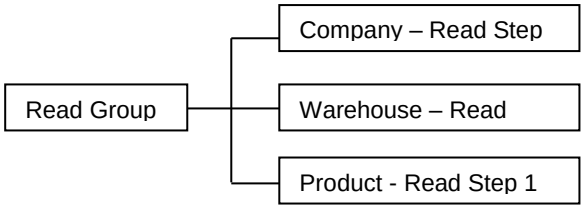
Read Step

Output Fields

A read step is used to define the position of the field in a multi-part key. Read steps allow the developer to define the field position in the key. This is useful where a read on a report is provided from multiple sources.

Eg.

Key = Company * Warehouse * Product



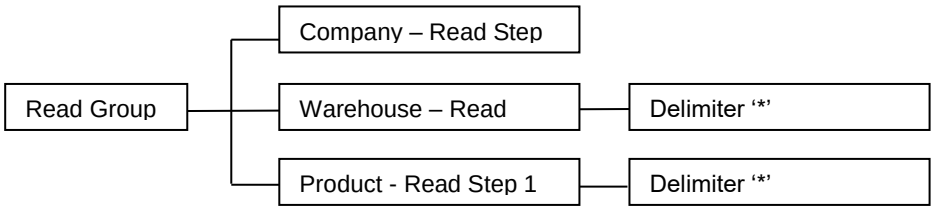
Delimiter

Output Fields

Establishes the delimiter to be used between each element of the key. Typically in a multi-part key the delimiter follows each key part with no delimiter set in the final key part.

Eg.

Key = Product * Warehouse * Company



Read to Variable

Output Fields

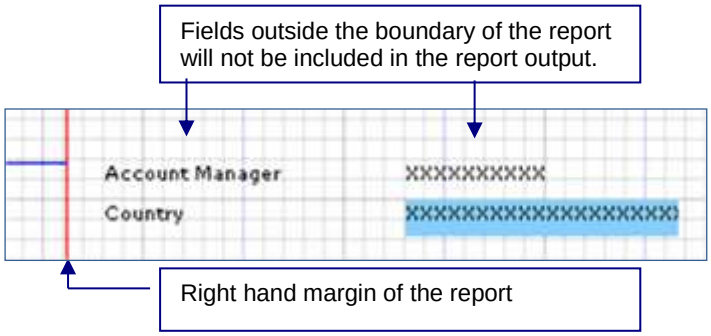
Is used to define which field DesignBais variable is used to store the results of the read.
Read Variables can only be used once on a report.

Valid values are:

DBOTHER.RECORD(1) – DBOTHER.RECORD(99)

Once read, the developer may add fields from the file that has been read. These fields must use the defined DBOTHER.RECORD subscript. The example below shows a read on the file DBCOUNTRY and the adding of the country name field to the report.

In the following example, the read of the DBCOUNTRY file is performed on an output field that appears outside of the boundary of the report. Fields outside of the report boundary do not print.



This technique enables the developer to simply add supportive reads to a report form. From the above example, the developer can add any of the fields from the DBCOUNTRY file to the report. DesignBais will automatically link the fields from the DBCOUNTRY file to DBOTHER.RECORD(4) read variable.

Controlling Report production through Subroutine Calls

In any of the 'Process' fields, the developer may enter either a valid subroutine name or C: for a Code Block. The subroutine must be catalogued.

Process Before Report (Main Report Form)

Will be invoked before the report is produced. Is used to do any preliminary processing before the report is produced.
PROCESS.EVENT will be set to "BEFORE REPORT".
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information

Process After Read (Main Report Form)

Will be invoked after the main read. This read will be assigned to DBRECORD and read group 1
PROCESS.EVENTSOURCE will be set to "REPORT READ"
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information

Before Read (Field Properties)

Output fields
Will be invoked before a read occurs. This will enable the developer to change key values before a read takes place.
PROCESS.EVENT will be set to "BEFORE READ"
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information

After Read (Field Properties)

Output fields
Will be invoked after a read defined on an output field.
PROCESS.EVENT will be set to "AFTER READ"
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information

Parameter

Output fields
Can be used by the developer to provide extra information to a subroutine being called in one of the 'Process' slots. The parameter being set will be placed in the variable PROCESS.PARAMETER.
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information

Derived in Subroutine

Output Fields.
PROCESS.EVENT will be set to "DERIVED"
Please see the section titled ***Tracking events and event processing*** in the ***Common variable usage and Subroutine interaction*** section of this manual for further information

ExportLink this field as the header to another field

This defines whether this field is linked as the header for another field. This field is only used during the spreadsheet transfer. By default, the spreadsheet export uses the header from the Field Property. If this field has a value it will overwrite this default behavior.

Report Header Text (Spreadsheet extract)

If specified, the text in this field will be used as the column header text in the spreadsheet transfer function. This is useful where the standard dictionary heading is not adequate for the spreadsheet.

Export Actions for Multi-values

The options are:

Leave as displayed.
Do not alter the display of other fields when current field is multi-valued

Normalise all columns to the depth of this field (MV Depth)
If this field contains multi-values, all other data fields on the report will be normalised to the maximum depth of the multi-value.

For example:

Name	Value
AAA	123
	456

Would become:

Name	Value
AAA	123
AAA	456

Flatten MV's and make new columns

This option will create a number new columns on the report. The number of new columns depend on the maximum depth of the multivalue field.

For example:

Name	Value
AAA	123
	456

Would become:

Name	Value.1	Value.2
AAA	123	456

This is very useful when creating tables to be used as input sources for mail-merges, whereby there are always defined columns for multi-value fields.

Change Multivalue to Spaces

Change all occurrences of a multivalue character to a space when exported to a spreadsheet.

For example:

Name	Value
AAA	123
	456

Would become:

Name	Value
AAA	123 456

Ignore this Field

There may be times where fields that are on the report are not intended to be exported to a spreadsheet. In this case, set this property to 'Ignore this Field'. This will ensure that the field is not added to the spreadsheet.

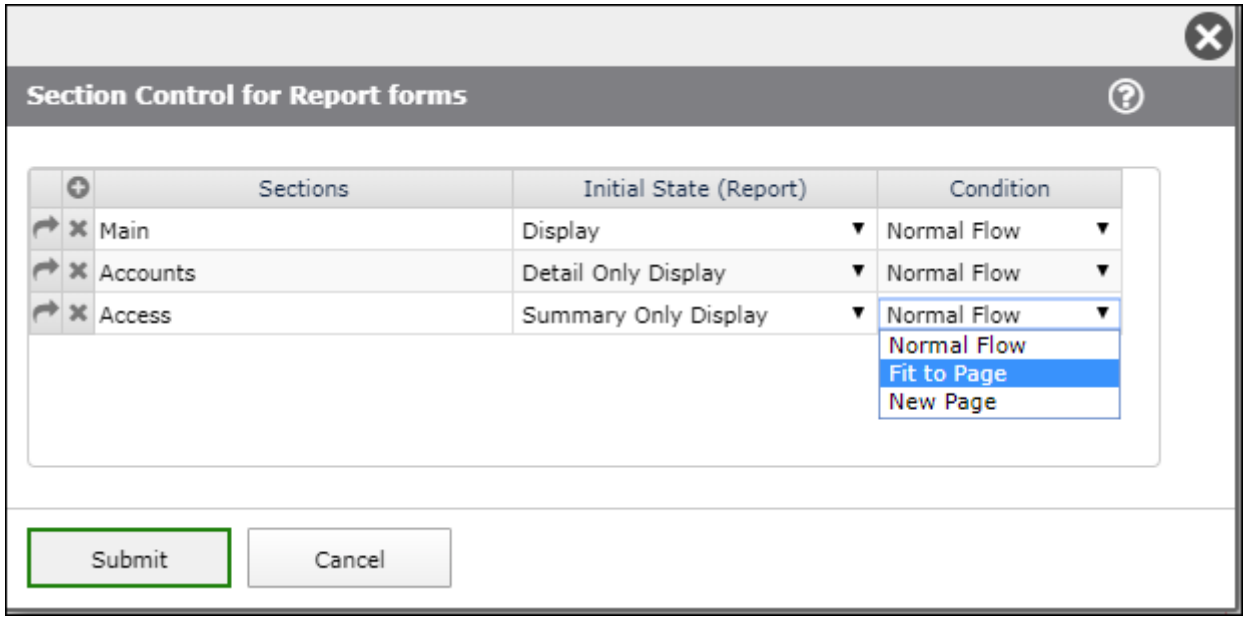
Change Multivalues to Line Feed

Change all occurrences of a multivalue character to a line feed when exported to a spreadsheet.

Sections (Menu Option)

This menu option is used to define the display state of each control of the form during runtime.

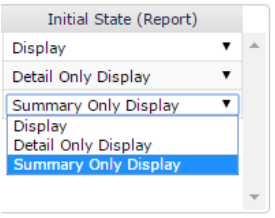
Report sections differ to forms sections in that they only pertain to controls being displayed in Detail or Summary Mode. You may assign controls on a report form as belonging to a section that will be displayed in either run state.



Prompts

Initial State (Report)

Used to define the display state of controls within each section on the report.



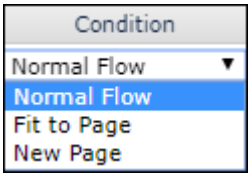
Display
Will display the controls belonging to the section in summary and detail modes

Detail Only Display
Will display the controls belonging to the section only in detail mode

Summary Only Display
Will display the controls belonging to the section when the report is run in summary mode

Condition

Used to control where the printing of a section containing variable depth fields will start.



The *Normal Flow* option is the default setting. Fields in the section will print immedialtely following any above fields in the report and if the content exceeds the page depth a new page will be triggered.

The *Fit to Page* option will attempt to place the content of variable depth fields on the current page but if there are not sufficient rows remaining on the page then printing of the section will start on a new page.

The *New Page* option forces the section to commence on a new page regardless of the number of rows remaining on the current page.

Buttons

Submit

Will commit the changes made.

Delete Field (Menu Option)

This menu option is used to delete the selected control/field on the report form. The currently selected field will be deleted.

Insert Row (Menu Option)

This menu option is used to insert a row (one grid cell in depth) to the report form. You must first select the Insert Row menu option and then click on the grid location where you want the row inserted. This function is repeatable until another menu option is selected.

Delete Row (Menu Option)

This menu option is used to delete a row (one grid cell in depth) from the report form. You must first select the Delete Row menu option and then click on the grid location where you want the row deleted. This function is repeatable until another menu option is selected.

Insert Column (Menu Option)

This menu option is used to insert a column (one grid cell in with) to the report form. You must first select the Insert Column menu option and then click on the grid location where you want the column inserted. This function is repeatable until another menu option is selected.

Delete Column (Menu Option)

This menu option is used to delete a column (one grid cell in width) from the report form. You must first select the Delete Column menu option and then click on the grid location where you want the column deleted. This function is repeatable until another menu option is selected.

Undo Last Action (Menu Option)

This menu option is used to undo the last action. The form will be reset as per the last action and any selected fields will be de-selected. If for some reason the developer crashes (as a result of an event subroutine call failure) this option can be used to take you back to last action of the form. When this occurs, select the 'Undo Last Action' option as the very first action when entering back into the forms designer. This will recover any lost actions up to the last action.

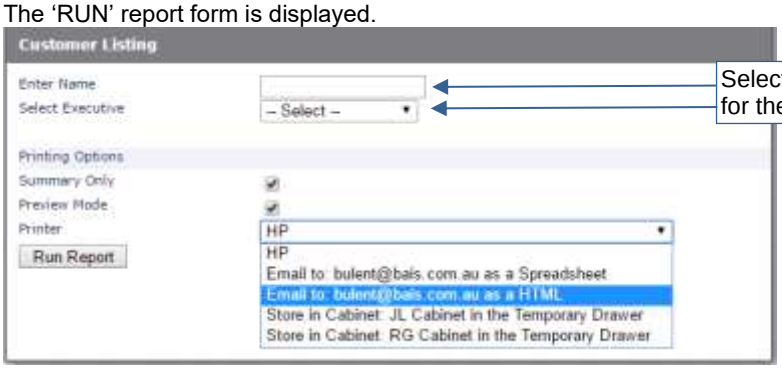
Review (Menu Option)

Opens the Form Fields display form. The developer can view selected field types. The Form Reads button displays a list of reads for all fields on the report.

Test Report (Menu Option)

Reports can be tested by simply selecting the 'Test Report' menu option.

The 'RUN' report form is displayed.



Prompts

Summary Only

Determines whether the report is run in detail mode or summary mode. Summary Mode removes the detail section from the report and only displays break fields.

Preview Mode

Determines whether the report is produced in preview mode or is sent directly to the printer.

Printer

Is used to select the printer to direct output to.
If the DesignBais user record has an email address attached, there will be two additional options available for the printing.

Email to: *email address* as a spreadsheet
Email to: *email address* as a HTML (if DBMail is not active)
Email to: *email address* as a PDF (if DBMail is active)

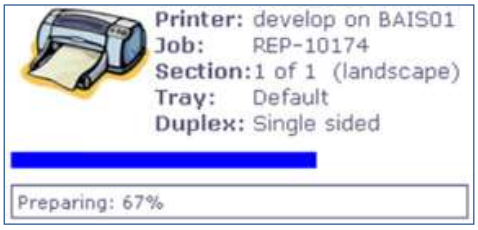
Both of these options will deliver the report (in the requested format) as an attachment to an email

Buttons

Run Report

Will invoke the report generator. At this point a phantom process is started to run the report. This avoids any web-server based time-outs.

When the report is running a progress window will be displayed.



If you have selected Preview, a different window will appear. Please see the Cabinets section in the System Parameters menu for more details.

Save (Menu Option)

This will save the form. It is prudent to save the form as you go.

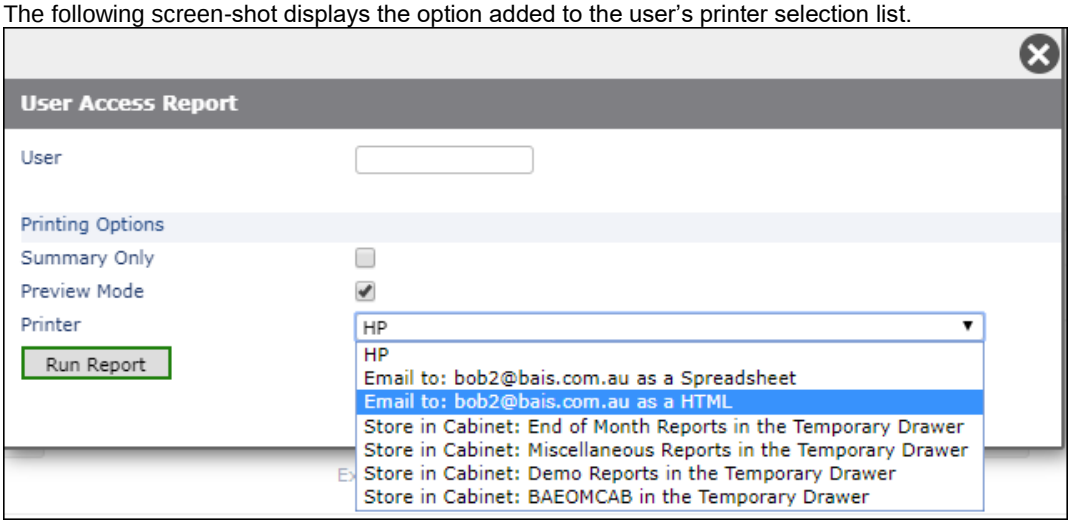
Exit (Menu Option)

This will exit the forms designer and return to the main first forms designer form.

Email and other report runtime properties

If a DesignBais user record has an email address entered the user will automatically receive the option to produce a report into a spreadsheet format.

The following screen-shot displays the option added to the user's printer selection list.



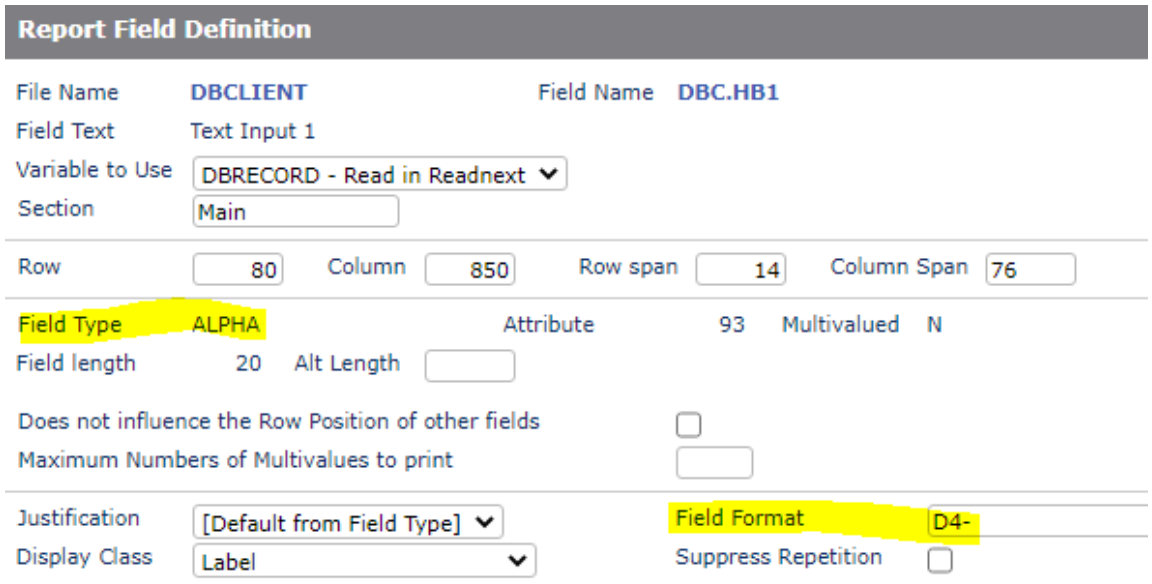
The above example displays the option to email the report in spreadsheet format to a selected email address. When a report is produced as a spreadsheet file, it is flattened. Any data cell on the report becomes a column in the resulting spreadsheet. Also, any break field will be inserted in the left-most columns in the spreadsheet. This ensures that all data (including break-fields) are added to the spreadsheet. Break-totals are not included in the spreadsheet. This reduces the likelihood of these figures causing confusion in the spreadsheet.

Exporting report data

When exporting data to a spreadsheet date fields are controlled by the dateFormat setting. The Field Format from the Report Field Definition (see below) or the Field Type from the Field Properties are used to set dateFormat.

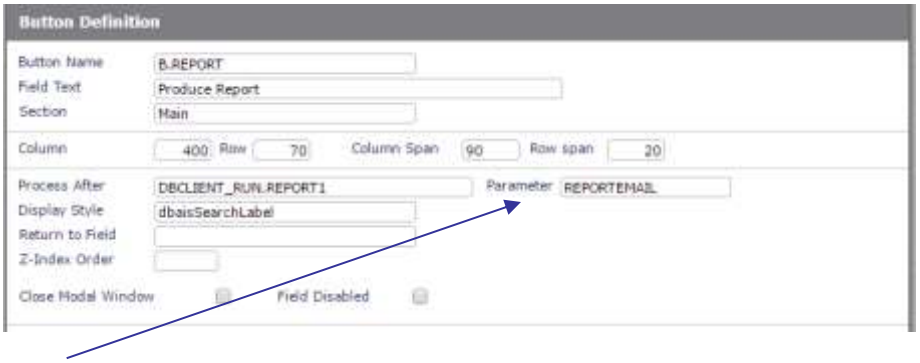
If the Field Type is *Date* then DesignBais will check the Field Format to allow for such things as *DJ* (Julian - day of the year only).

Hence, an Alpha field with a Field Format that matches the stored string will present as a date in eXcel.
If the Field Format matches the data but there is a different date format then the field is sent to eXcel as a string.



Report Run Time Options

If the user has used the "Do Not Display Run Window" when designing the report, there are a number of run-time parameters that can be used from a button to direct output correctly.



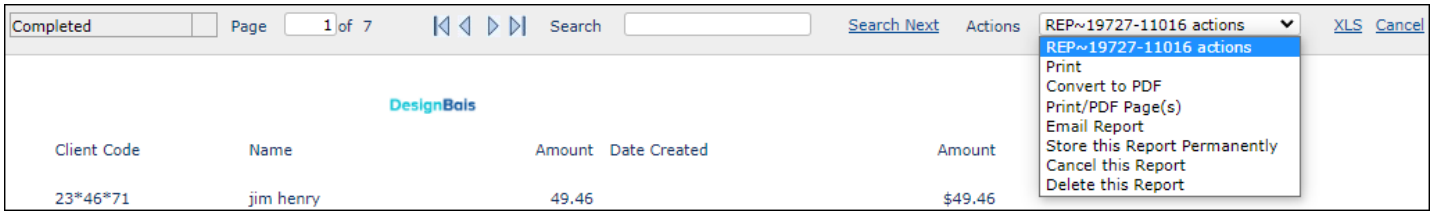
Parameter

REPORTEMAIL	Directs output to the email address supplied in user record, sending a proper Excel file if DBMail is running
REPORTPRINT	Directs output to non-preview print
REPORTHTML	**(**was **HTMLEMAIL)** Directs output to the email address as a HTML or PDF depending on whether DBMail is active.
REPORTCSV	Option to email the report data as a CSV. Double quotes in the data replaced with single quotes, data elements are enclosed in double quotes & separated by commas.

Report Run Time Actions on Page Control Form

The Actions dropdown list includes the following:

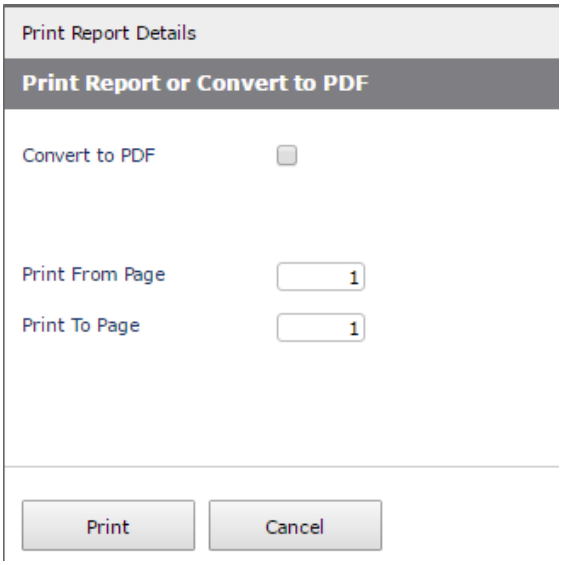
- REP~19727-11016 actions
- Print
- Convert to PDF
- Print/PDF Page(s)
- Email Report
- Store this Report Permanently
- Cancel this Report
- Delete this Report



REP~19727-11016	This is part of the key of the record on the &PH& file (on Universe) holding the contents of the report. The full key has the time and date appended separated by underscores. For example REP~17840-10008_56035_17840.
Print	This will print the report on the selected printer.
Convert to PDF	Creates a PDF of the report in the /uploads folder on the Web Server. The file is assigned a random unique key.

Print/PDF Page(s)

This allows the user to print selected pages of the report, and optionally, to create a PDF.

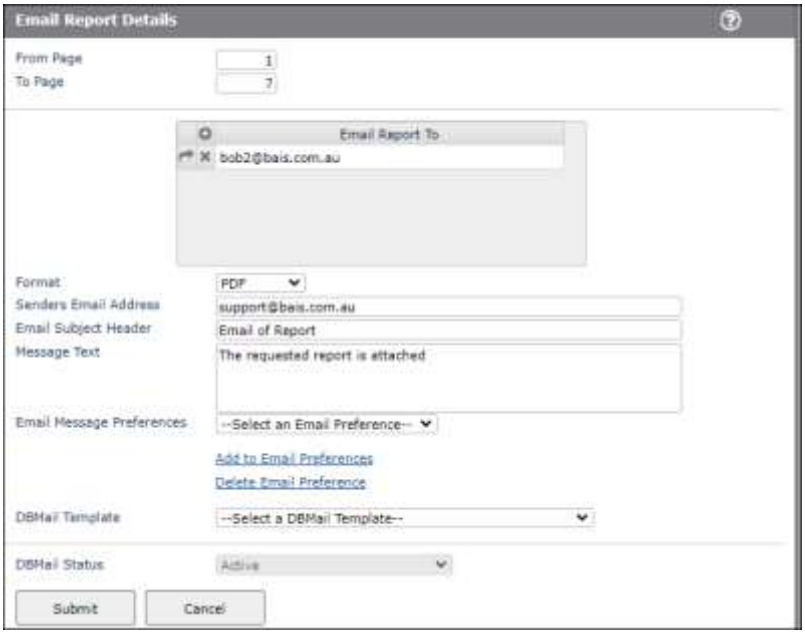


Email Report

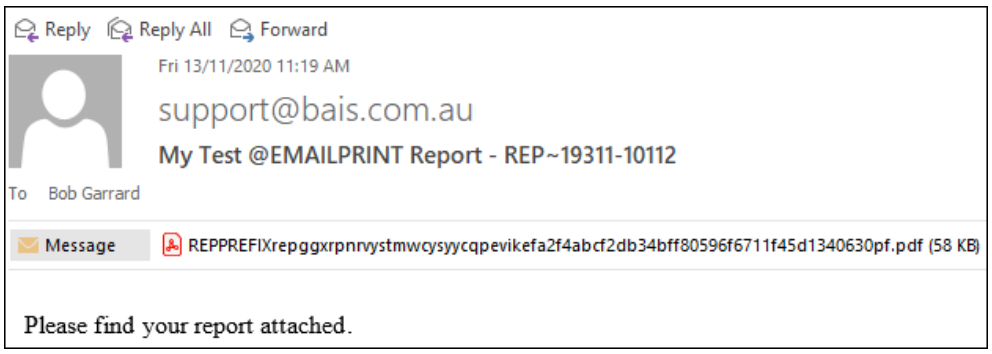
The default *Email Report To* address is that of the user running the report. This is the equivalent of the *Email Report to Self* option that existed in DesignBais prior to Release 8.6.0.1.

If DBMail is running then a PDF of the report is sent to the email address held on the DBIUSERS file if the default address is accepted, or a PDF is sent to the list of email addresses entered in the *Email Report To* grid.

Use the *Format* dropdown selection to send either a PDF, an Excel file, or CSV data file. The CSV option is equivalent to the *Email Report CSV Data* option that existed in DesignBais prior to Release 8.6.0.1.

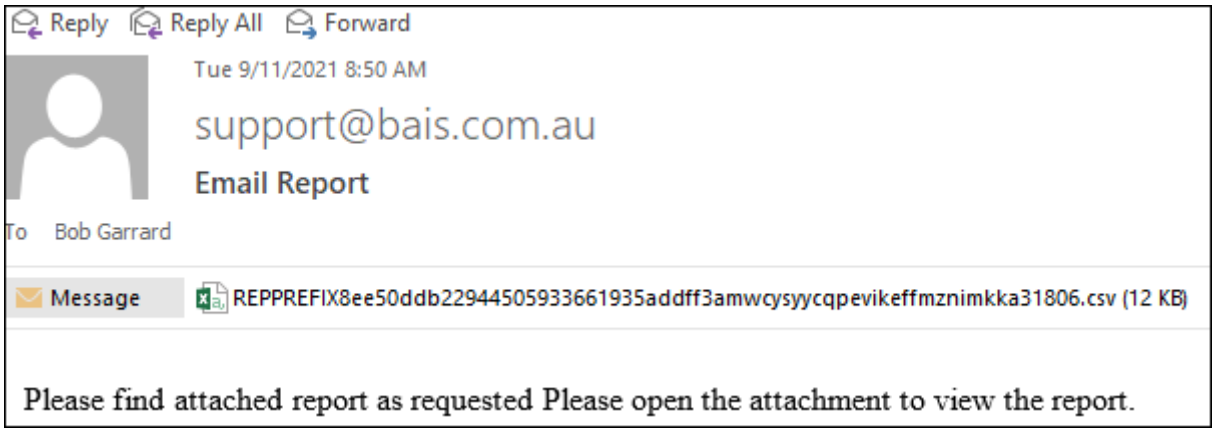


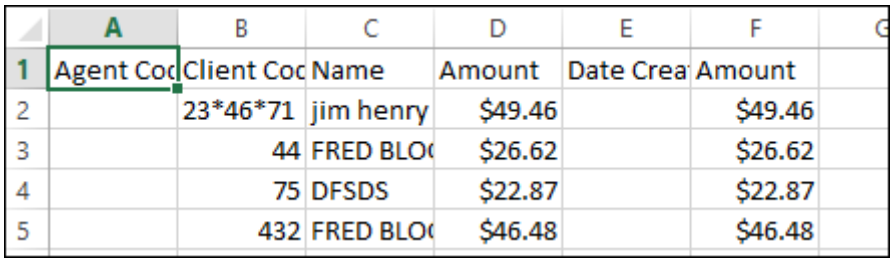
Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3



If DBMail is not active then a link to the /uploads folder is emailed to the specified email addresses.

If the *CSV Only* option is selected then an email similar to the following is sent:

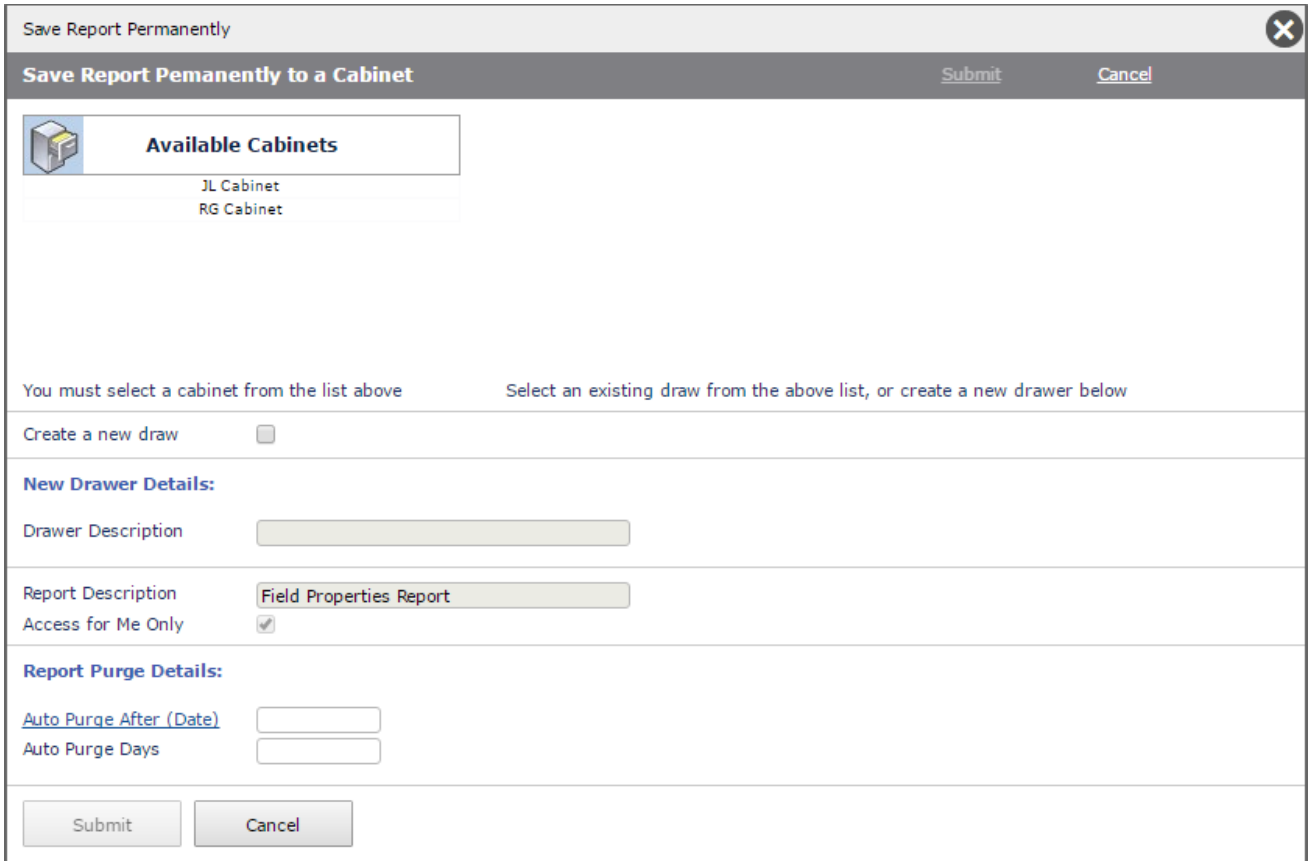




	A	B	C	D	E	F	G
1	Agent Coc	Client Coc	Name	Amount	Date Crea	Amount	
2		23*46*71	jim henry	$49.46		$49.46	
3		44	FRED BLO(	$26.62		$26.62	
4		75	DFSDS	$22.87		$22.87	
5		432	FRED BLO(	$46.48		$46.48	

Store this Report Permanently

This option allows the user to save the report to an eXpress Reports Cabinet.



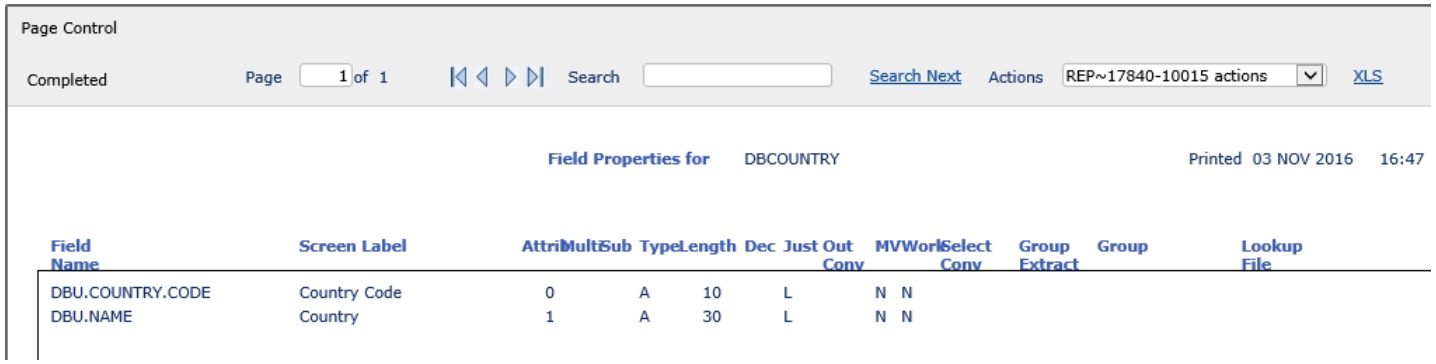
Cancel this Report — This cancels the report. It is equivalent to the Cancel hyperlink in the header row of form M31.

Delete this Report — This will delete the Report from the eXpress Cabinet.

Note that this does not remove the record from the &PH& file. This file must be maintained/purged separately.

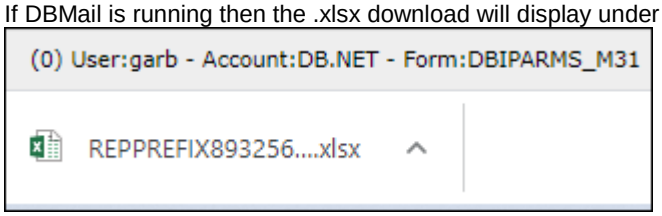
The /uploads folder must also be maintained separately.

XLS Button on the Page Control Form

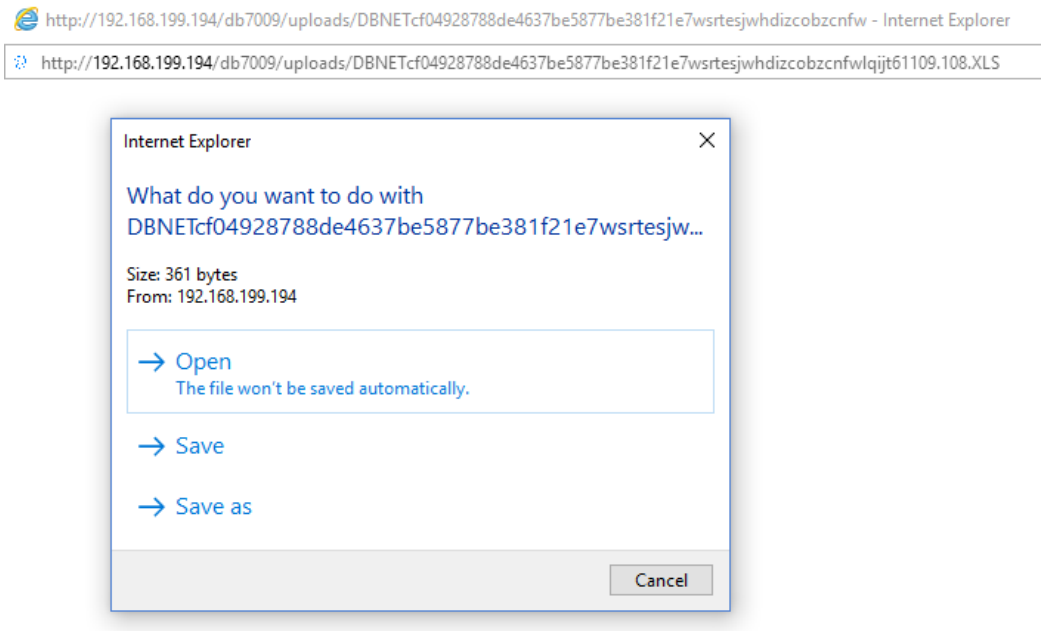


XLS — This creates a .xls spreadsheet file of the report.

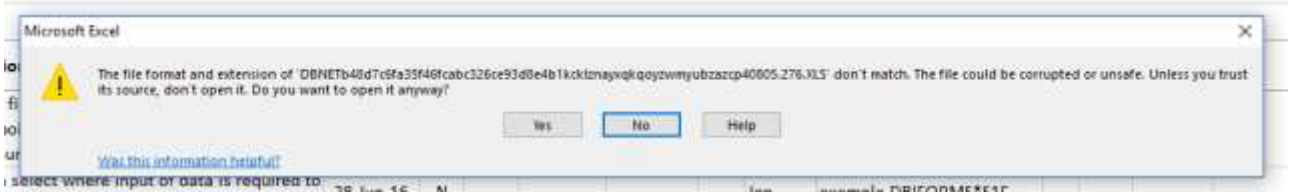
If DBMail is running then the .xlsx download will display under the task bar:



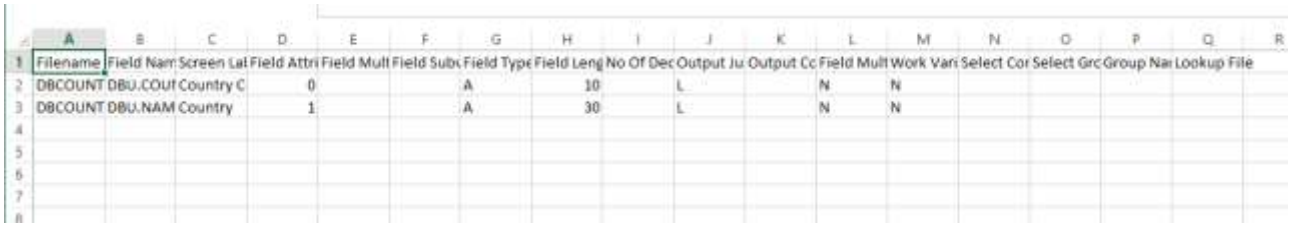
Otherwise in Internet Explorer the following is displayed:



In Chrome:

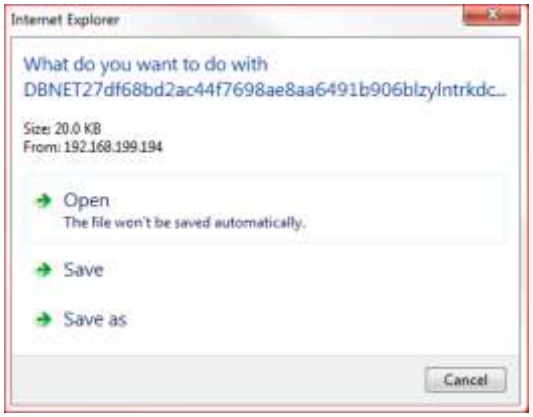


The file is created in the /uploads folder. This folder is on the web server. Move the folder to your C drive and then open it with Excel. The report is loaded into Excel rows and columns corresponding to the report rows and columns.

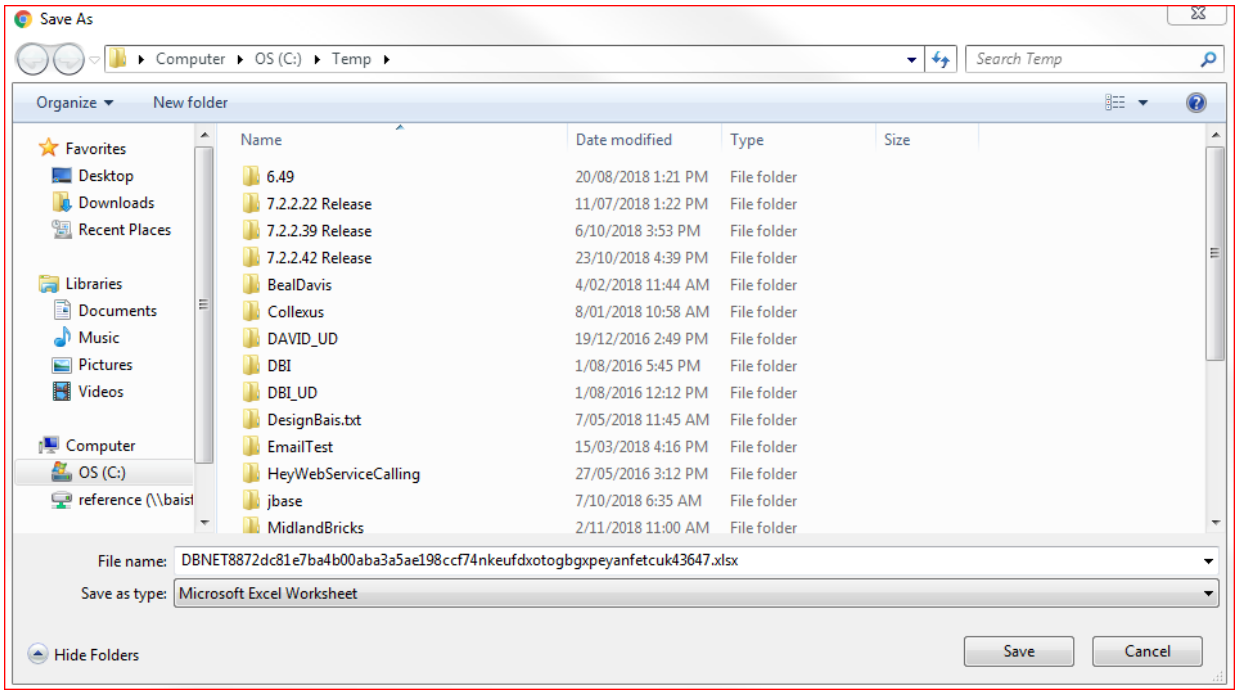


Note that the blank browser window opened by the XLS button in which the download tab appears will remain open after the downloaded file is accessed. Clicking the XLS button will not recreate the download file until this browser window is closed. Note also that the Excel format warning may be hidden by an already open Excel sheet. Click the task bar Excel icon to show all open Excel sheets and the warning message will be visible.

In Internet Explorer the following is displayed:



In Chrome:



In Edge:

The way that DesignBais opens Excel files is now:

In IE and Edge, the Excel file is directly opened in a new window. The status bar etc won't be missing and the window will be positioned where it should be.

In other browsers, the file will be downloaded at the bottom of the existing window instead of opening a blank browser window.

Hiding Fields in DesignBais Reports and Excel Export Files

Use DBSECTIONSPEC to hide selected sections of a report.

Usage:

DBSECTIONSPEC = Section Name *VM* {Section Name}
DBSECTIONSPEC<2> = Change to State

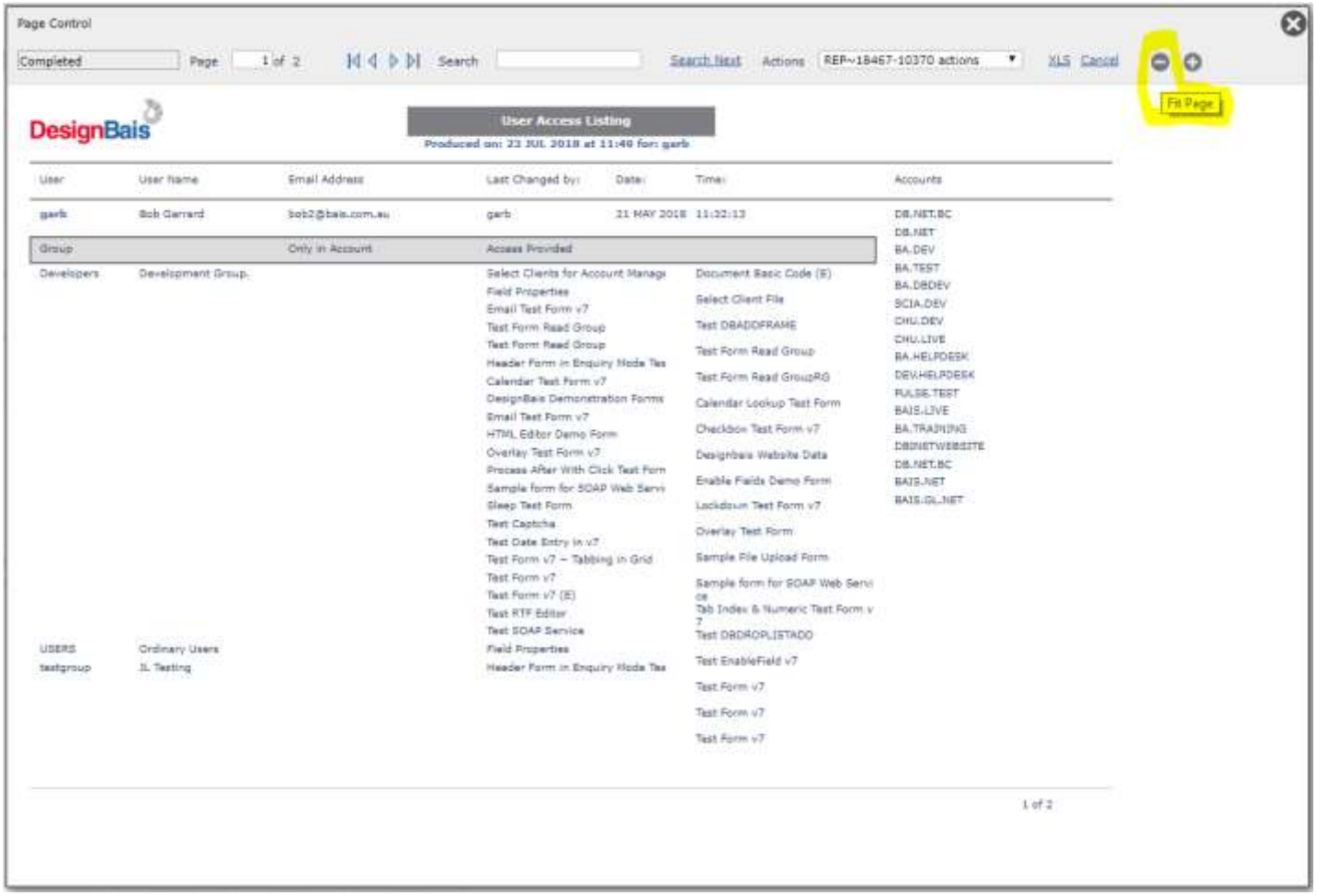
The valid Parameters for the Change to State are:

H	Hidden
HE	Hidden on report and on excel export file

Zoom Buttons on the Page Control Form

The "-" and "+" buttons allow the user to zoom in to and out of the displayed report to display the entire page and remove the vertical scrollbar.

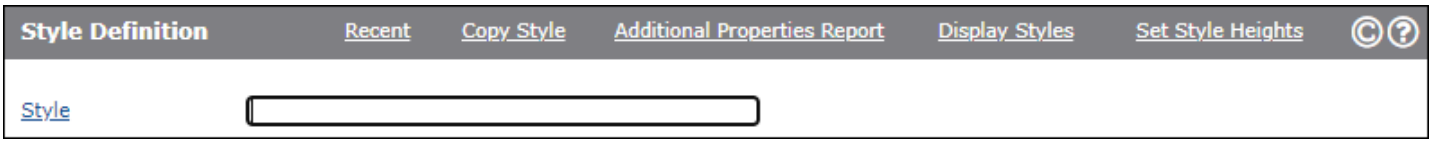
The "-" reduces the page size so that the scrollbar is not required. The "+" restores the full size display.


Page Control
Completed
Page 1 of 2
Search
Search Next
Actions
REP~18467-10370 actions
XLS
Cancel
Fit Page
DesignBais
User Access Listing
Produced on: 23 JUL 2018 at 11:49 for: garb
User
User Name
Email Address
Last Changed by:
Date:
Time:
Accounts
garb
Bob Garrard
bob2@bais.com.au
garb
21 MAY 2018
11:32:13
DB.NET.BC
DB.NET
BA.DEV
BA.TEST
BA.DBDEV
SCIA.DEV
CHU.DEV
CHU.LIVE
BA.HELPDESK
DEV.HELPDESK
PULSE.TEST
BAIS.LIVE
BA.TRAINING
DBDNETWEBSITE
DB.NET.BC
BAIS.NET
BAIS.GL.NET
Group
Only in Account
Access Provided
Developers
Development Group.
Select Clients for Account Manage
Field Properties
Email Test Form v7
Test Form Read Group
Test Form Read Group
Header Form in Enquiry Mode Tes
Calendar Test Form v7
DesignBais Demonstration Forms
Email Test Form v7
HTML Editor Demo Form
Overlay Test Form v7
Process After With Click Test Form
Sample form for SOAP Web Servi
Sleep Test Form
Test Captcha
Test Date Entry in v7
Test Form v7 – Tabbing in Grid
Test Form v7
Test Form v7 (E)
Test RTF Editor
Test SOAP Service
Document Basic Code (E)
Select Client File
Test DBADOFRAME
Test Form Read Group
Test Form Read GroupRG
Calendar Lookup Test Form
Checkbox Test Form v7
Designbais Website Data
Enable Fields Demo Form
Lockdown Test Form v7
Overlay Test Form
Sample File Upload Form
Sample form for SOAP Web Servi
ce
Tab Index & Numeric Test Form v
7
Test DBDROPLISTADO
Test EnableField v7
Test Form v7
Test Form v7
Test Form v7
USERS
Ordinary Users
testgroup
IL Testing
Field Properties
Header Form in Enquiry Mode Tes
1 of 2

Chapter 10 – Styles/Style Groups

Style Definition

This option allows the developer to add or change the default display properties of text label and input controls on the form.



Options in the Label Header bar:

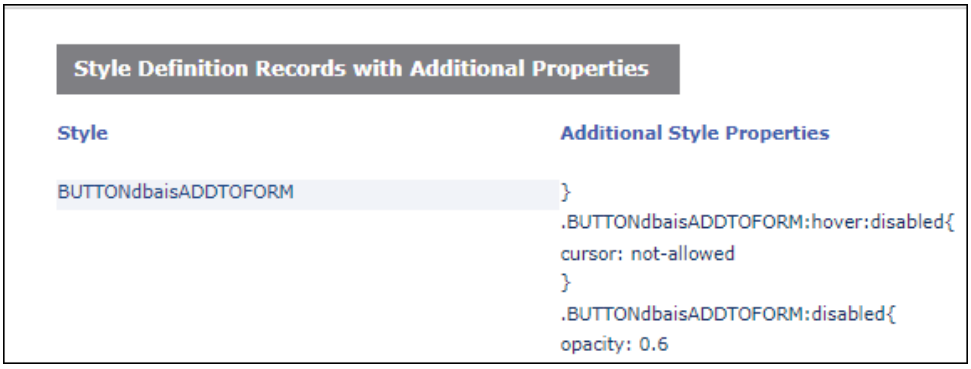
Recent — Display a list of recently maintained styles allowing reselection.

Copy Style — Copy an existing style definition to a new name. Select either Style Definition to select a style from the local DBISTYLE file, or System Styles to select a style from the DesignBais styles records in DBISYSFORMS. In the latter case select the required style to copy from the System Style Name dropdown at the base of the form. The *Style Name* defaults to the style being edited.

See section below *Copying and Renaming Styles.*

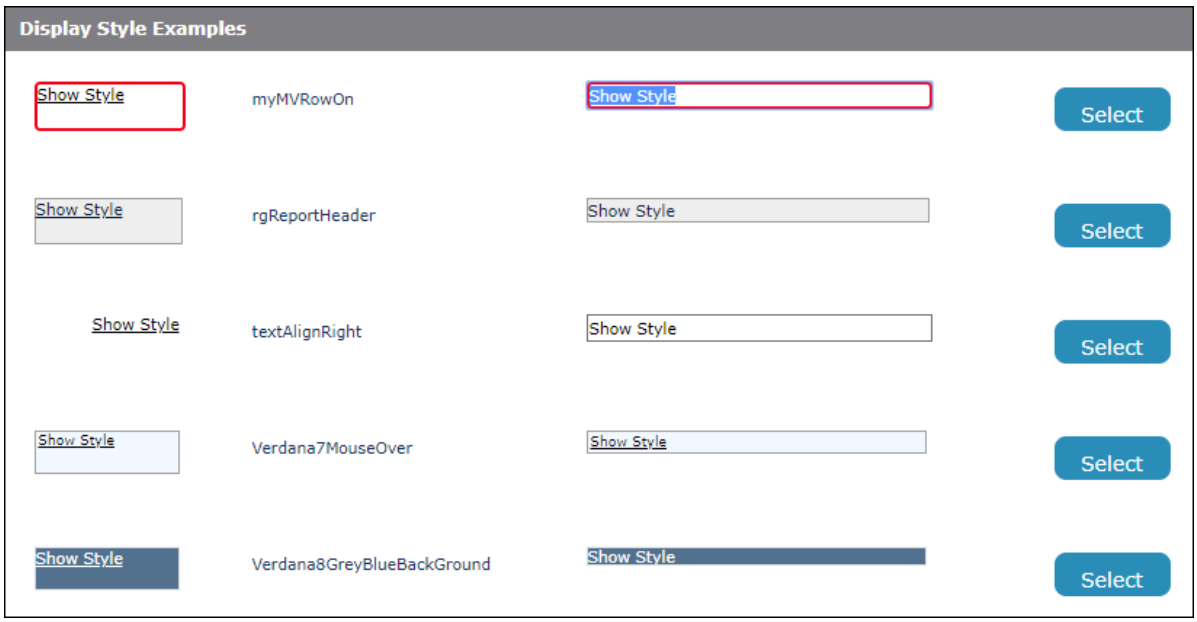
Additional Properties Report

A report to display all rows of the Additional Properties field for ease of reference.

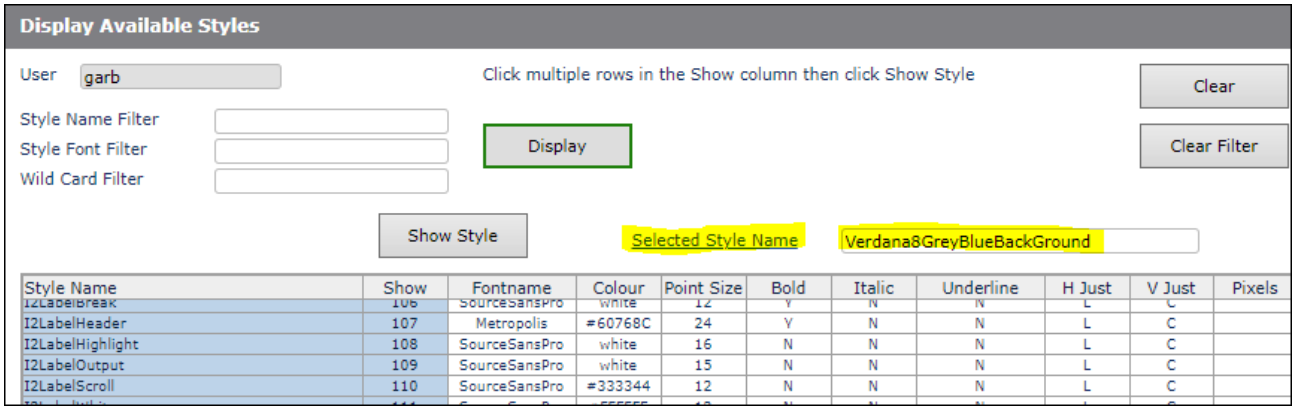


Display Styles — Opens a form that provides the option to display a filtered selection of styles using any of the Style Name Filter, Style Font Filter or Wild Card Filter. From this display you can select any number of rows by clicking in the *Show* column.

Clicking the *Show Style* button builds a form that displays each of the selected styles as applied to a button and an input field. This permits the developer to see what a style looks like before applying it to a form.



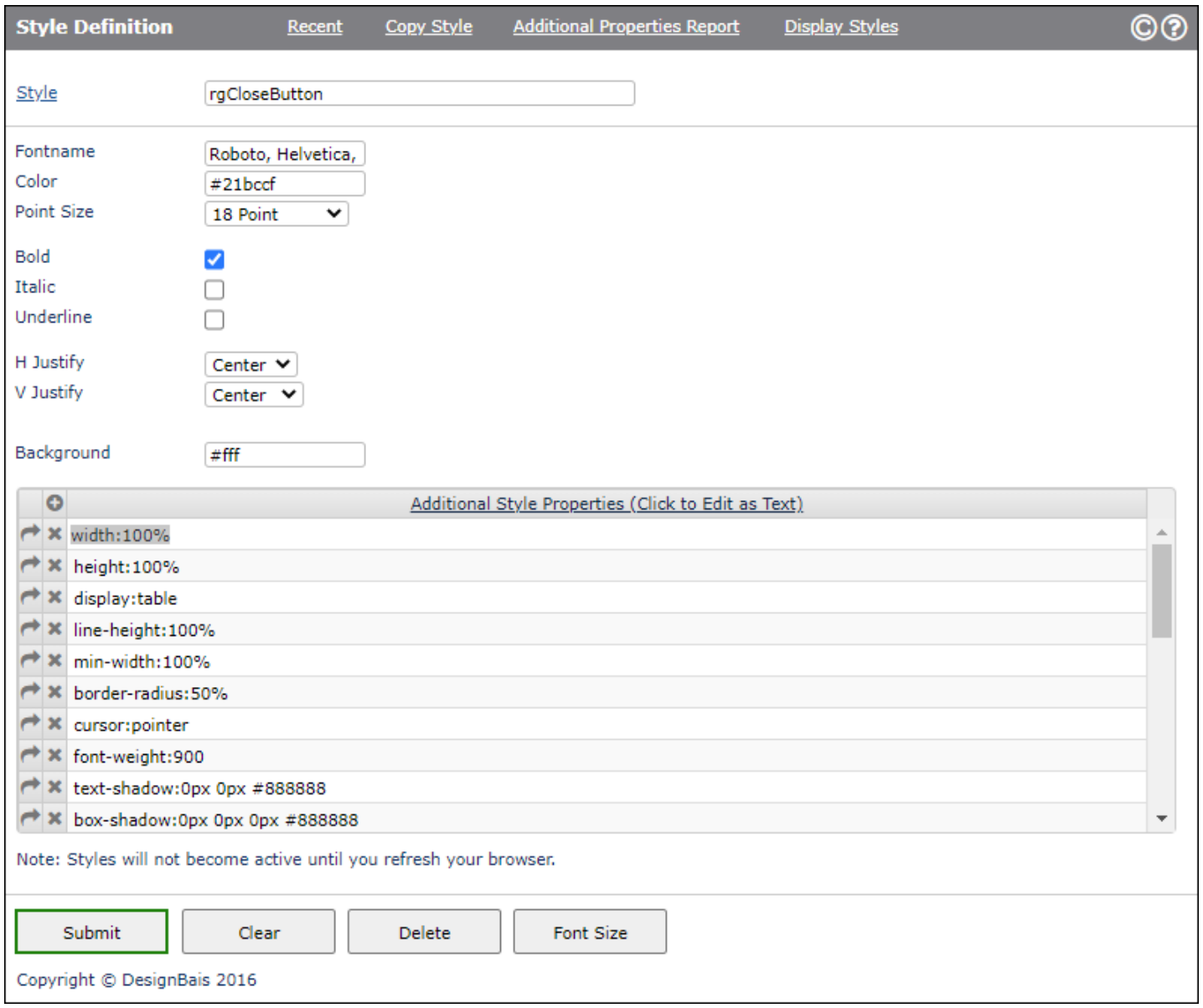
Click the blue *Select* button on the right to select a particular style. This style name will then be loaded into the *Selected Style Name* field. Click this link to display the style properties.



Set Style Heights

Builds a list of all *Font* and *Font Size* combinations from the DBISTYLE style definitions. DesignBais calculates the width of all printable characters for normal, italic, bold and bold italic, and row height for a Font and Point Size combination. The table of calculated values is stored on DBIPARMS in records keyed by *FONT*fontname*pointsize.* These calculations are used for printing and for building selection and run report forms.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3



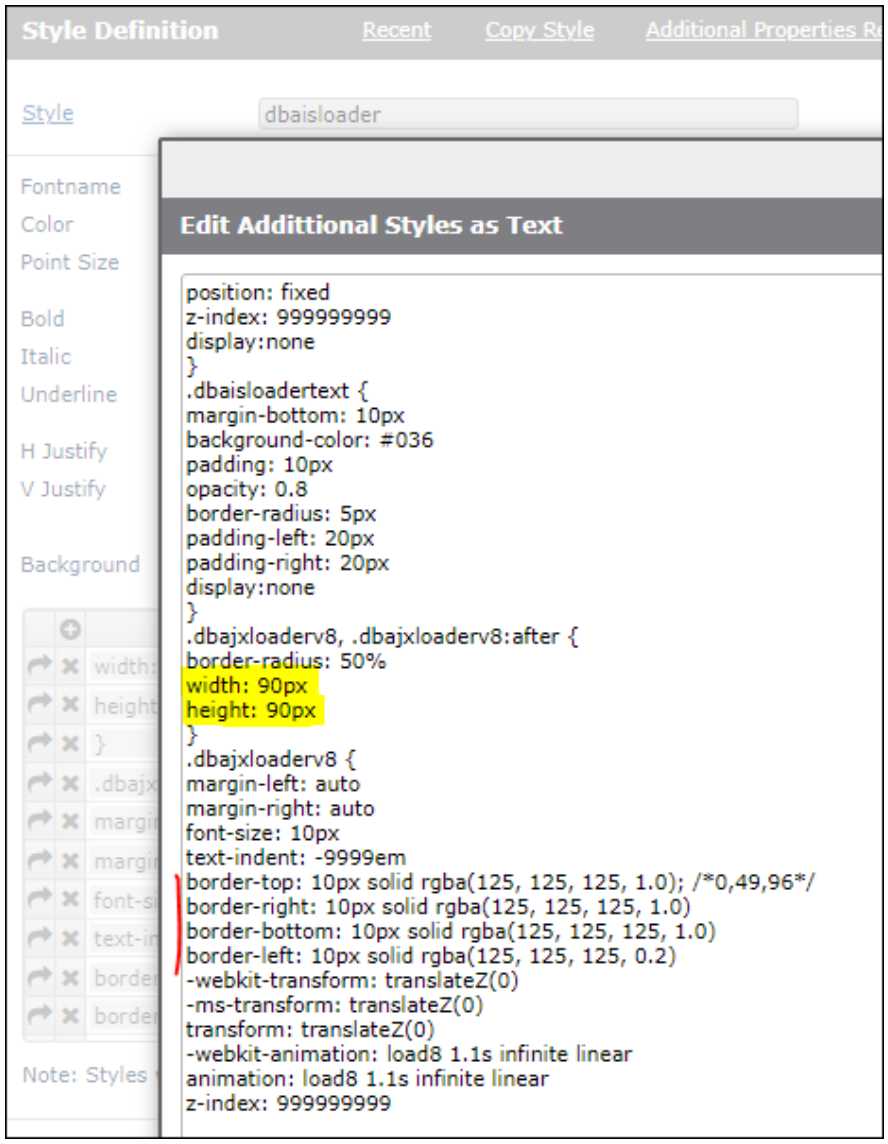
Style Definitions are linked to a Style Group to provide a consistent look and feel to the application. DesignBais ships with 2 standard Style Groups dbaisWeb and dbaisRep.

The developer can change some basic parameters (e.g. color) of styles within these groups but making changes to other attributes of the style may cause unpredictable results.

For all styles with 'dbais' as part of the style name only the following properties should be changed:

- Color
- Background (color)
- Fontname

The exception to this is the style 'dbaisloader'. This style is maintainable by developers. This style controls the look of the DesignBais spinner (the icon that appears during a server hit).



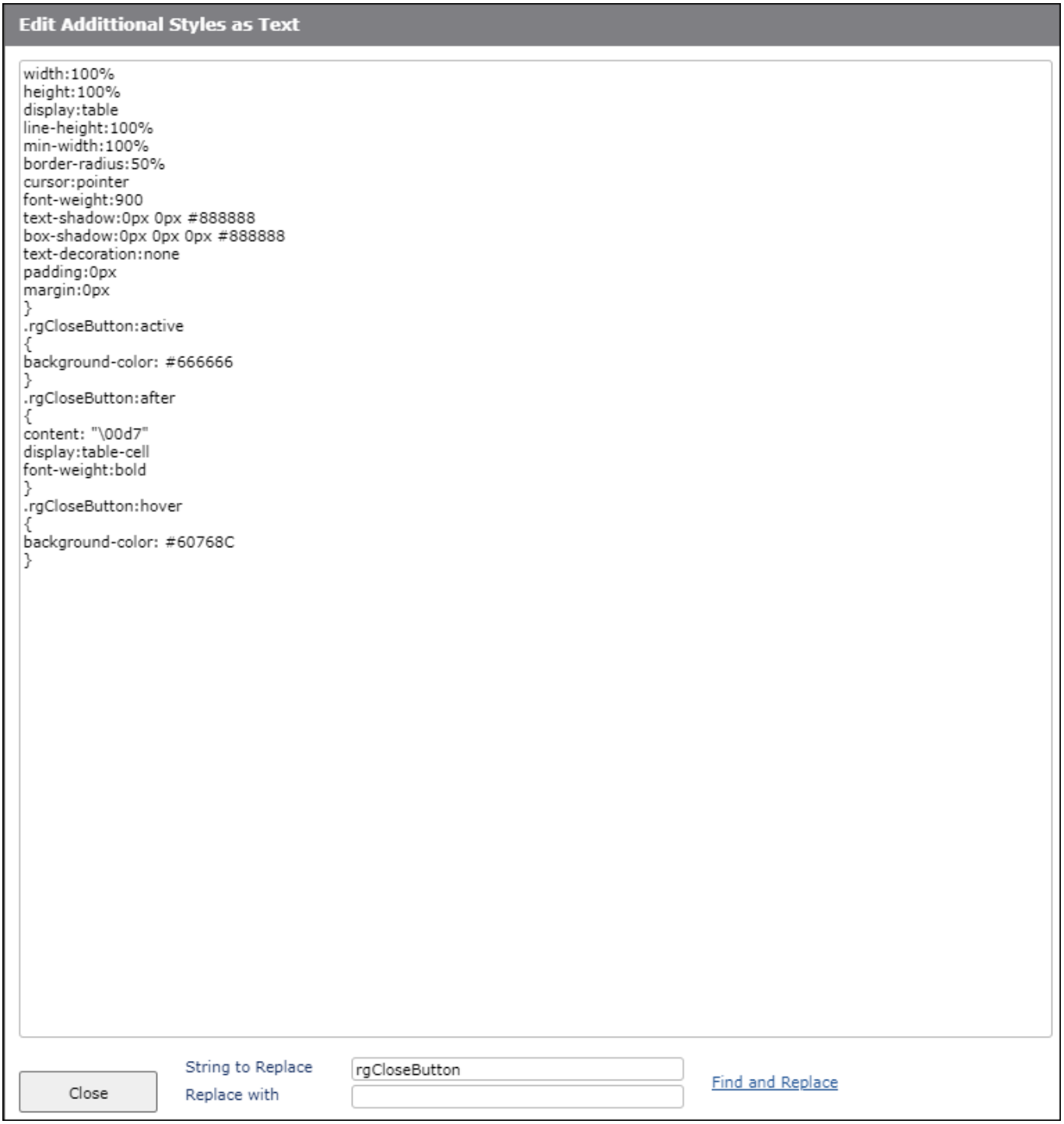
To change the spinner size adjust *width* and *height* (marked in yellow).

To change spinner width and color adjust the *border-top right bottom left* attributes (marked with red).

You may change the look of the spinner by adding lines to the additional styles in the *dbaisloadertext {}* section. Note that color and other attributes are inherited so to change the color, for example, add a line like: *color: red*

Prompts

Style	Is the name given to the style definition.
Fontname	The name of the font to be used for this style. DesignBais recommends the use of the Verdana font for all form-based fonts. It is the cleanest font at varying font sizes. When assigning a font for printer output, Arial, Times New Roman or Courier New, are probably the best choices. It is important to assign a font that most printers will support. Verdana, for example is not normally a supported printer font.
Color	Can be either one of the 140 color names for browsers, or an RGB color.
Point Size	Defines the size of the font being used. 8 point is the recommended font-size for most form-based fonts.
Bold	Will display the font in bold
Italic	Will turn the font to italic mode
Underline	Will underline the font
Horizontal Justification	Will change the horizontal justification assigned to the style. Left is the default for most fields. This should only be changed when the developer wishes to center justify headings.
Vertical Justification	Will change the vertical position of the text within the label. This should always be set to Top.
Background	Will set the default background color for the style. Can be either one of the 140 color names for browsers, or an RGB color.
Additional Style Properties (Click to Edit as Text)	This multivalue field allows the developer to add various CSS properties to the style. These are entered by using braces to close the current style and start a new one – note the final closing brace is inserted by the submit logic. By using CSS we avoid the need for javascript in various events. Click the header to open a new form allowing the content to be edited as text.


Edit Addittional Styles as Text
width:100%
height:100%
display:table
line-height:100%
min-width:100%
border-radius:50%
cursor:pointer
font-weight:900
text-shadow:0px 0px #888888
box-shadow:0px 0px 0px #888888
text-decoration:none
padding:0px
margin:0px
}
.rgCloseButton:active
{
background-color: #666666
}
.rgCloseButton:after
{
content: "\00d7"
display:table-cell
font-weight:bold
}
.rgCloseButton:hover
{
background-color: #60768C
}
Close
String to Replace
rgCloseButton
Replace with
Find and Replace

Buttons

Submit — Will commit the changes and update the file **DBISTYLE.**
DesignBais will also update the CSS file immediately. The changes will not take effect in your current session until you refresh your browser as the browser caches the style sheet at session start-up.

Clear — Will clear the form and not commit any changes made.

Delete — Will delete the currently viewed style from the **DBISTYLE** file.

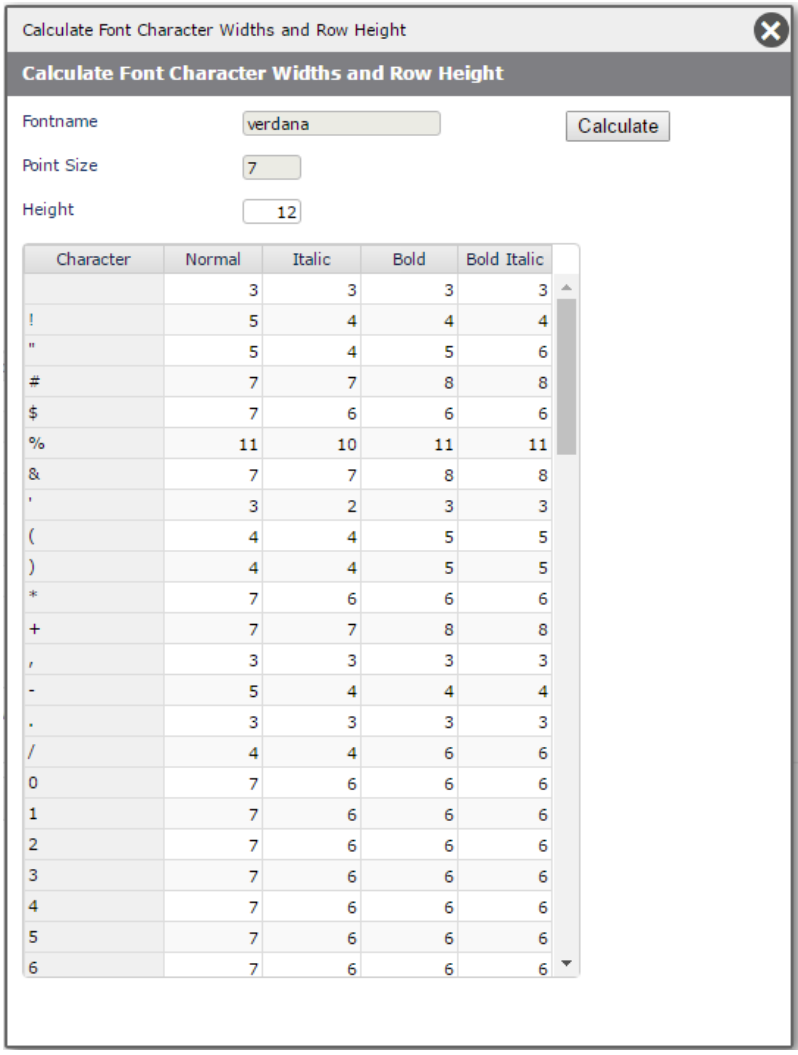
Font Size — Click this button to open the Calculate Font Character Widths and Row Height form. This form allows the developer to calculate display width and height for the full character set for the current browser. The calculated values are stored on DBIPARMS with a record id of "FONT*fontname*pointsize".

Each record is stamped with the browser type used for the calculations, such as Chrome or Firefox. This is important because the calculated widths are used when determining the col span of text fields and hyperlink buttons within Forms Designer and Report Designer.

Different browsers will render form and report elements differently. A column span setting that correctly contains the text for a label in one browser may be too short for another browser, such that he label breaks into two or more lines.

Developers of cross browser applications must keep this in mind.

DesignBais has found that of the common browsers (IE, Chrome, Firefox) Firefox requires longer column spans than the others in order to render a tag or hyperlink button name correctly. It is recommended that the Font Size calculation is run while using Firefox as this will ensure that a tag or hyperlink will render in the same way in all the commonly used browsers.



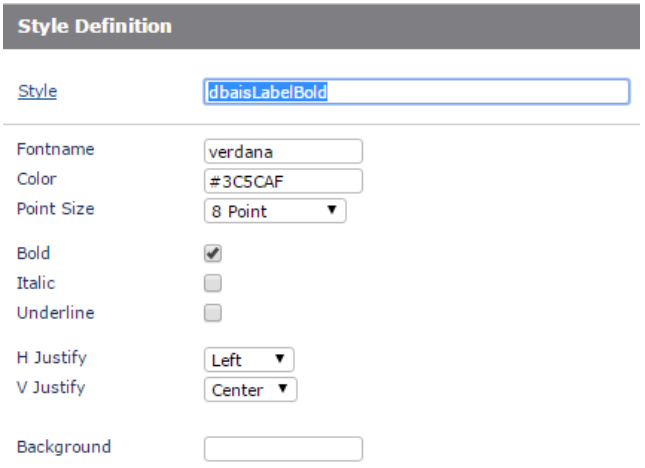
Character	Normal	Italic	Bold	Bold Italic
	3	3	3	3
!	5	4	4	4
"	5	4	5	6
#	7	7	8	8
$	7	6	6	6
%	11	10	11	11
&	7	7	8	8
'	3	2	3	3
(	4	4	5	5
)	4	4	5	5
*	7	6	6	6
+	7	7	8	8
,	3	3	3	3
-	5	4	4	4
.	3	3	3	3
/	4	4	6	6
0	7	6	6	6
1	7	6	6	6
2	7	6	6	6
3	7	6	6	6
4	7	6	6	6
5	7	6	6	6
6	7	6	6	6

DesignBais delivers 40 or so font calculations which are stored in the DBISYSFORMS file. These have been calculated using Firefox. DesignBais will reference these from within Forms and Report Designer when assigning the default col span for form and report elements.

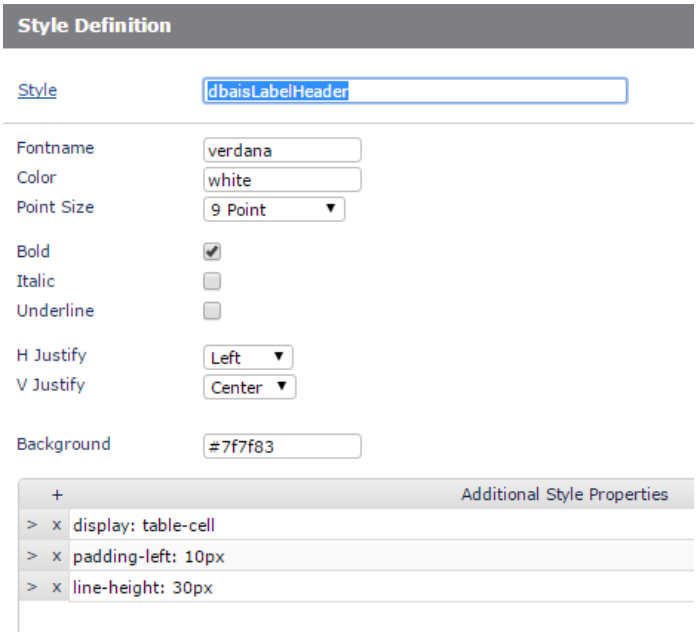
If the font and point size are not found in DBISYSFORMS then DesignBais will search for your FONT* records in DBIPARMS.

Style Examples

Creating a Bold style:



Creating a Header Style:



Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Styling Disabled Text Fields

From Release 8.6.0.6 DesignBais includes a dbaisState field in *Text* fields on a form so that they can be styled when disabled.

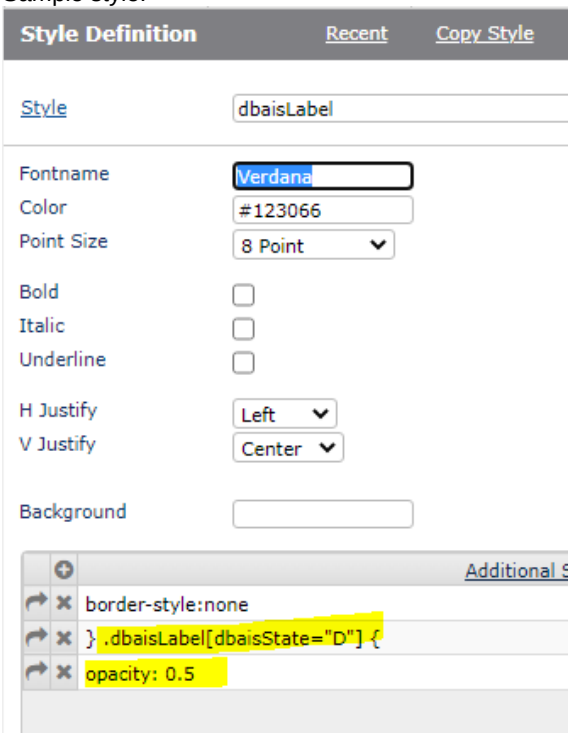
Use *dbaisState="E"* when the section is enabled
Use *dbaisState="D"* when the section is disabled

Using the css selector *.stylename[dbaisState="D"]* in your Style Definition will allow styling.

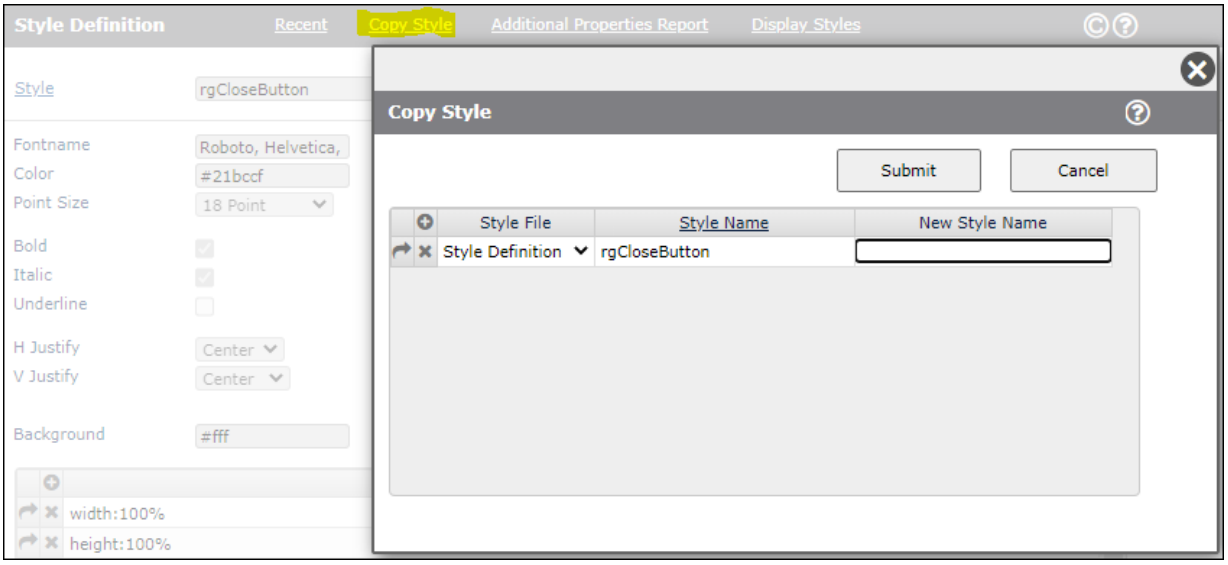
DesignBais sets the attribute dbaisState on labels.

In the snip below replace *dbaisLabel* put your style name.

Sample style:

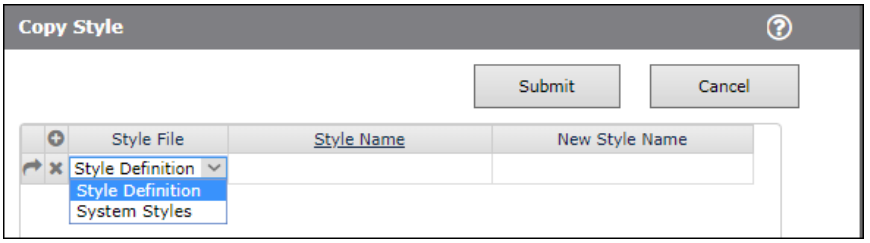


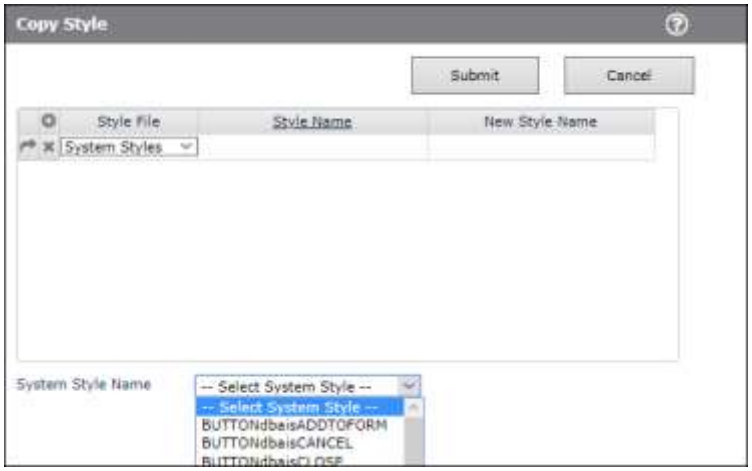
Copying and Renaming Styles



If you need to set up your own style definition based on a standard DesignBais definition then the procedure to follow is:

- In the Style Definition tool enter the name of the DesignBais style
- Click Copy Style in the header bar
- Select from the dropdown either "Style Definition" or "System Styles" (the former means copy a style definition from your local DBISTYLE file while the latter will copy a style from those supplied by DesignBais held in DBISYSFORMS).





- Enter the DesignBais style name, it defaults to the style entered on the main Style Definition form, then the name to which you want to copy the style definition
- Note that you cannot use a name containing the reserved DesignBais prefix "dbais"
- Click Submit
- You will be prompted to refresh your browser in order to regenerate the css file

When a style, containing its style name embedded in the Additional Style Properties, is copied then all occurrences of the style name are replaced with the new style name.

Changing the style of a MV Grid element

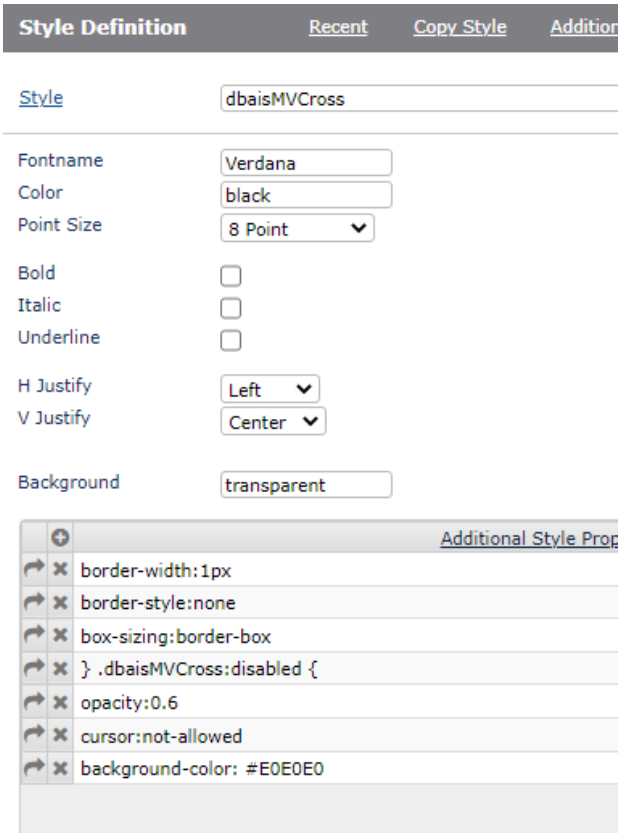
This example demonstrates a method to make disabled MV grid fields stand out more clearly.

M97	2809871	0	0	0
M97	2809872	0	0	0
M97	2875487	0	0	0
M97	2875488	0	0	0
M97	A4722032744	0	0	0

MV grids are a little complex with <input> elements inside <td>. You may have to style all of the components - the <tr>, the <td> and the input element.

Inspect the element to be changed to find the class. This will be a style from your DBISTYLE file or it could be a DesignBais style held on DBISYSFORMS.

In the style definition define a “disabled” setting as shown in this figure.



The syntax is important:

- You need to add (on grid line 4) the closing brace to end the base style.
- Do not enter a trailing closing brace on grid line 7 since DesignBais will add it automatically.
- Precede the style name with a “.” followed by “:disabled”.
- The open brace on grid line 4 then starts the new style.
- Add any css you want to be applied. Be careful with elements that change the size as it can affect the whole row.

Using Custom Fonts

DesignBais use “awesome fonts” in Release 7/8 by having the font in a site folder along with a css file and link to the font via a html insert with a class. For example the MV grid control to add a row uses classes fa and fa-plus-circle:

<td id="mvA_ASSOC1z1" onclick="vs(event)" class="dbaisMVHeader" style="cursor: pointer; width: 16px; overflow: hidden; border-style: solid; box-sizing: border-box; max-width: 16px; min-width: 16px;"></td>

If the requirement is to use a more standard font then this can be done as follows:

- The Style Definition form allows you to define the font in a DesignBais class element.
- The Style Groups form allows you to set up a set of classes for a report.
- The Report Designer lets you assign classes from the Style Group nominated to the report elements.
- DesignBais does a rough pt to mm conversion in order to translate to the physical paper which may affect the results.

To use custom fonts on a report, such as FontMICR for bank checks, you will need to set up the following:

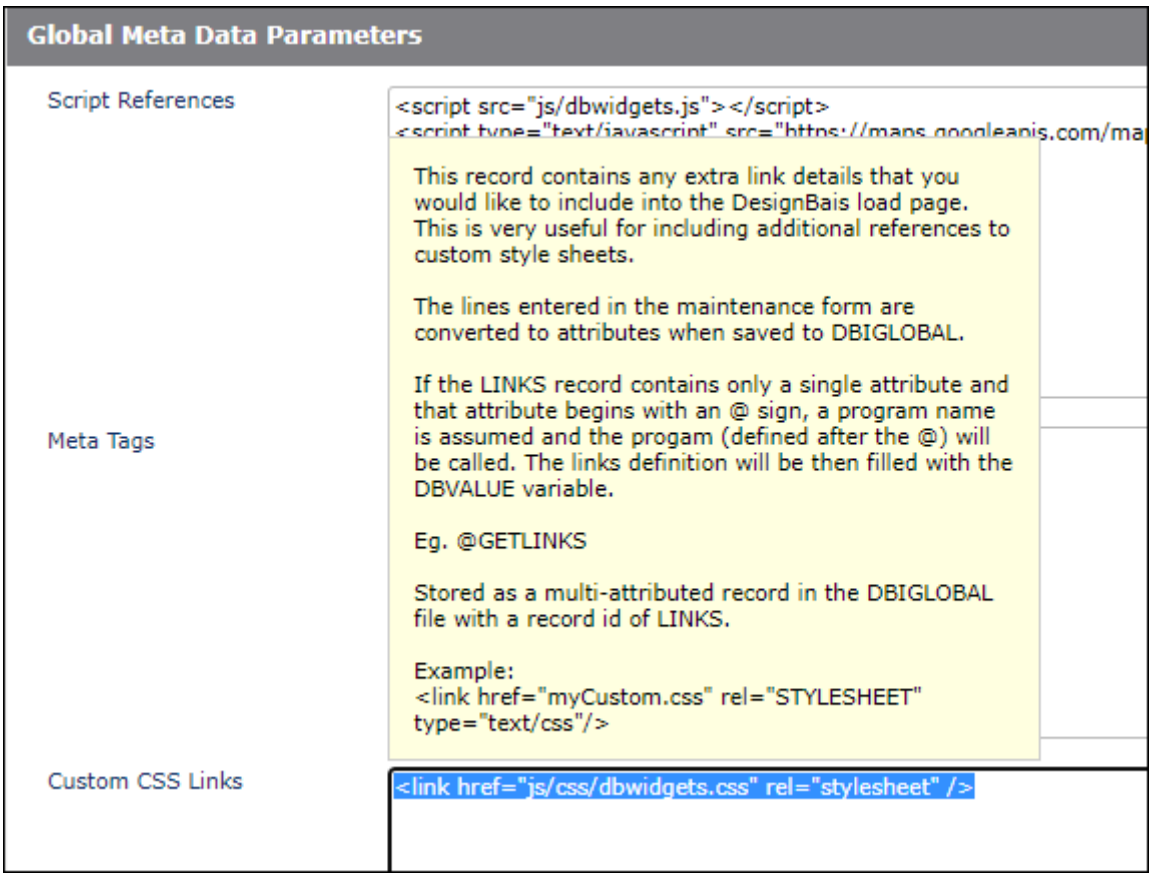
```
@font-face {
   font-family: 'YourFontName'; /*a name to be used later*/
   src: url('http://domain.com/fonts/font.ttf'); /*URL to font*/
}

.classname {
   font-family: 'YourFontName';
}
```

After that you can use that "classname" in your form design. The above definitions need to be in your DesignBaisStyle.css. They get there by creating styles in the Style Definition tool, clicking the submit button to save them and then refreshing the browser.

Note that the @font-face must appear in the css before the font is used.

The styles to be used on forms are not sorted in DesignBaisStyle.css but added via a simple basic ‘SELECT DBISTYLE’.

To get the @font-face command into DesignBais before DesignBaisStyle.css you will need a DBIGLOBAL file with a “META” item.



The Meta Tags field should contain a line like: <link href="css/myFonts.css" rel="STYLESHEET" type="text/css"/>
You may need to create the css folder on your website along with the file myFonts.css.

The file myFonts.css will contain the @font-face commands as described above.

You can then enter a “Style Definition” in the normal fashion referencing the new font remembering to refresh your browser afterwards.

Note that you do not need the leading “.” in the Style Definition name as it is added in the after write.

DBIGLOBAL is an optional file that was introduced to hold parameters across a number of accounts. It is recommended to create the file in the DBILOGIN account with Q-pointers from any data accounts much the same as DBIUSERS. However, it may be located anywhere in the database file system.

The DesignBais release accounts contain the DBIGLOBAL file from Release 7 onwards. The DICT file lives in the DBI account or DBINET in Release 7/8. In D3 the DICT lives with the data and is populated from the DBI/DBINET account.

In the earlier versions of DesignBais items are added using an editor but in Release 7/8 maintenance forms have been provided.

Once you have edited the META record you will need to refresh your browser as the link will be added to the <head> tag as the first DesignBais page is loaded.

Global Parameters -> Custom Fonts creates a DBIGLOBAL FONTFACE record.

This is currently added at the start of the DesignBaisStyle.css as an @font-face{font-family: … ;src= … ;}.

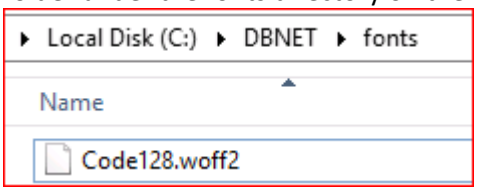
After adding a Custom Font a dummy amend on a style (or create one using the new font-family defined) is needed to re-create the stylesheet. Naturally a browser refresh is also needed.

The @font-face must be placed before the @media print element in the DesignBaisPrint.css so that the custom font will also work in Internet Explorer and FireFox when printing a report.

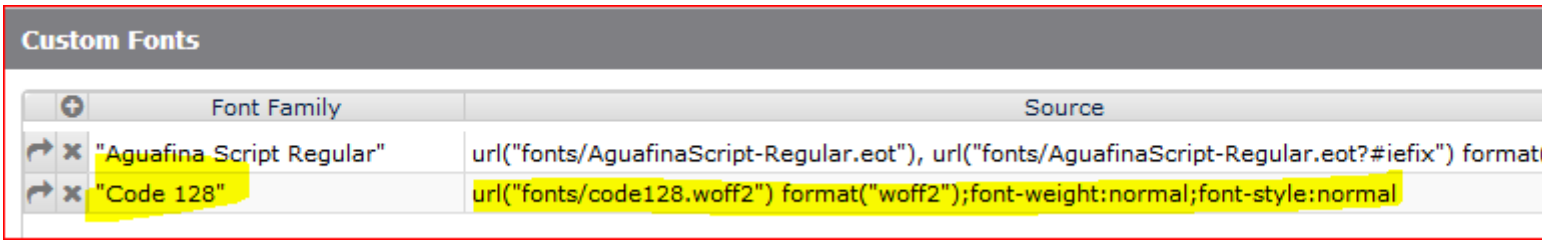
Using Custom Fonts for Barcode Printing

Find a suitable barcode font on the web. In this case a code 128 barcode font is used.
http://www.barcodelink.net/barcode-font.php to download code128.ttf
Use https://transfonter.org/ to convert to woff2. This is not essential. Your browser of choice may not support woff2.

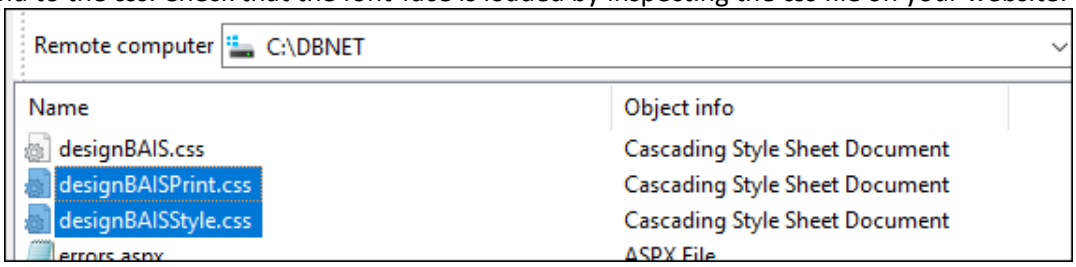
Place the converted font file in the *fonts* folder of your website. It is recommended that you place it in an application folder under the fonts directory on the website:



Add a reference to the font file to the Global Parameters Custom Fonts list:



Dummy amend a style to trigger the re-build of the DesignBais style sheet. This will add the new custom font-face command to the css. Check that the font-face is loaded by inspecting the css file on your website:

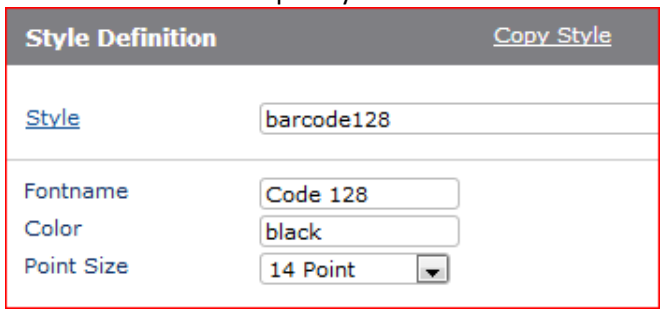


DesignBaisStyle.css

```
@font-face {
font-family: "Code 128";
src: url("fonts/code128.woff2") format("woff2");
font-weight:normal;
font-style:normal;
}
```

Repeat this procedure for the *DesignBaisPrint.css*.

You then need to set up a style to define the barcode font:

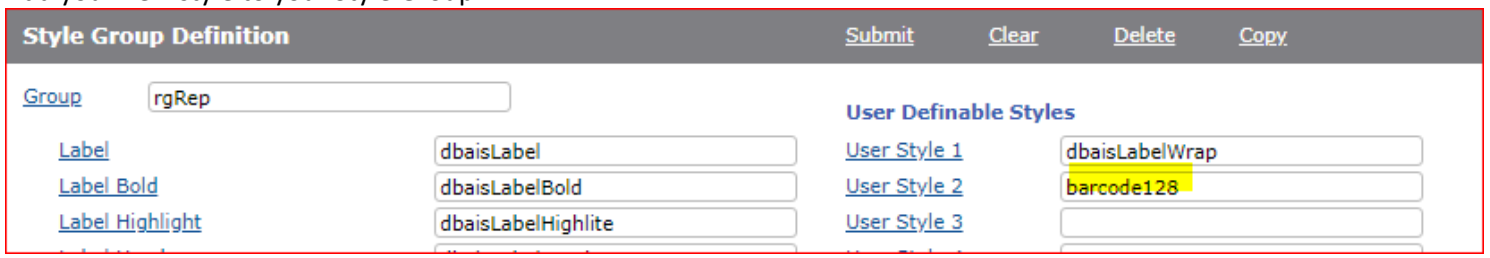


Refresh your browser.

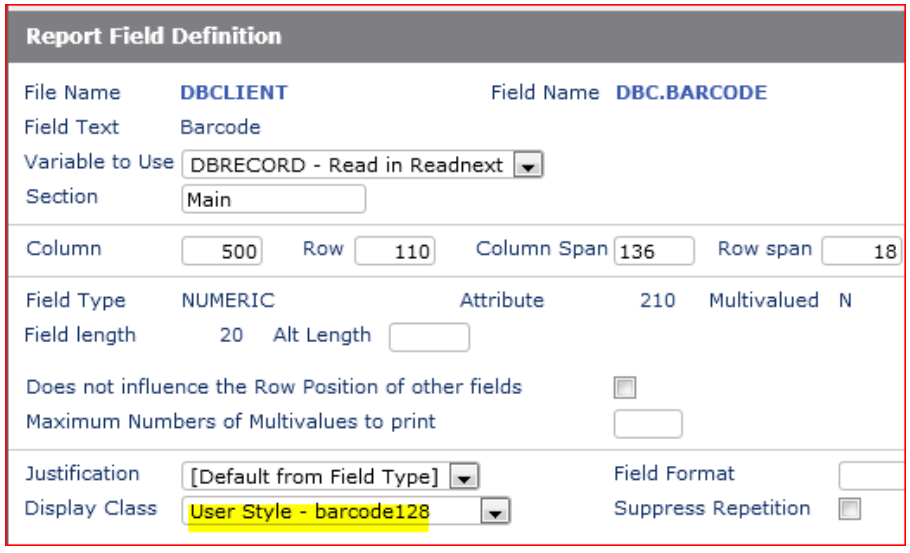
DesignBaisStyle.css

```
.barcode128{
font-family: "Code 128";
font-size: 14pt;
color: black;
vertical-align:middle;
}
```

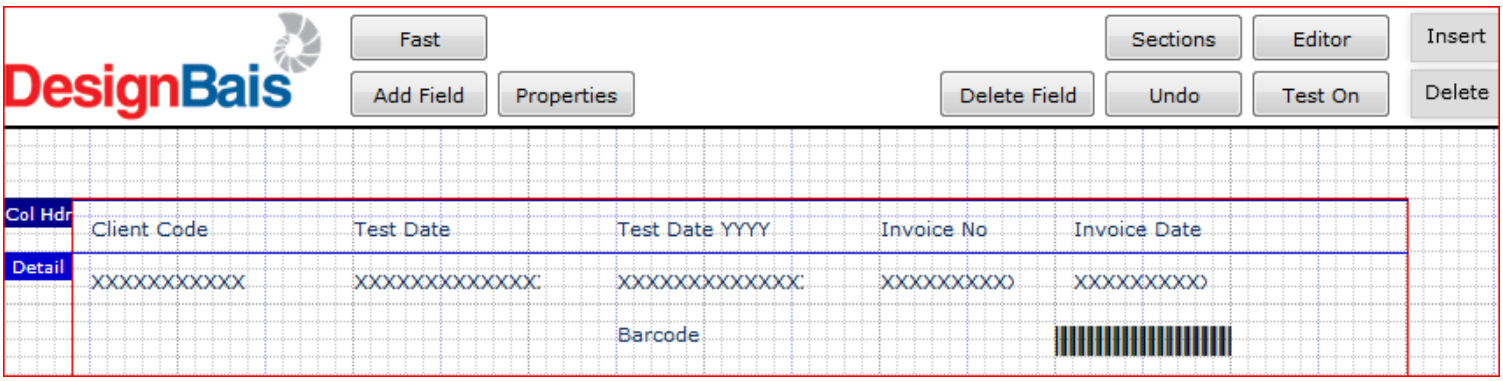
Add your new style to your Style Group:



Add a field to your report and set the Display Class to use the new font – in this case *User Style 2*:



Run the report to check the barcode:



Style Overflow

DesignBais allows developers to control the text overflow of a field. By default DesignBais will contain the text within the element except in the case of the header row in on-form reports where overflow is the default. If, however, the string *overflow* is found in the style record that applies to the field then DesignBais will not add *overflow:hidden* to the xml string. The code snippet below clarifies DesignBais action in relation to white-space, overflow and word-wrap. A value in HDRIDNAME indicates that the element is an on-form report header.

```
*
* Check Style Insert for white-space, overflow & word-wrap
*
      IF WS.REQ AND INDEX(STYLE.INS,'white-space',1) = 0 THEN
         IF HDRIDNAME = '' THEN
            STYLE.INS := 'white-space:nowrap;'
         END ELSE
            STYLE.INS := 'white-space:normal;'
         END
      END
      IF OF.REQ AND INDEX(STYLE.INS,'overflow',1) = 0 THEN
         STYLE.INS := 'overflow:hidden;'
      END
      IF WW.REQ AND (INDEX(STYLE.INS,'word-wrap',1) = 0 AND INDEX(STYLE.INS,"overflow-wrap",1) = 0) THEN
         STYLE.INS := 'overflow-wrap:break-word;'
      END
      RETURN
```

If a DesignBais field displays unwanted overflow then the developer should check which style is being used and check if overflow is enabled in the style record. Remove any reference to *overflow.*

Developers should be aware that *overflow-hidden* is not valid in DesignBais 7/8. The correct syntax is *overflow: hidden*.

DesignBais Styles

Styles used by the DesignBais tools have been established to ensure correct display of the various elements used in DesignBais forms. These styles cannot be amended using the tools. If they are copied then the following should be kept in mind.

The following style sheets are included with DesignBais:

- jquery.alerts.css
- compatible w3c.css
- awesome.css

```
<link href="jq/jqalerts/jquery.alerts.css?v=57" rel="stylesheet" type="text/css"> == $0
<link href="designBAISStyle.css?v=57" rel="stylesheet" type="text/css">
<link href="designBAIS.css?v=57" rel="stylesheet" type="text/css">
<link href="css/compatible_w3c.css?v=57" rel="stylesheet" type="text/css">
<link href="css/awesome.css?v=57" rel="stylesheet" type="text/css">
<link rel="stylesheet" href="jq/themes/redmond/jquery-ui.min.css?v=57" type="text/css">
<link rel="shortcut icon" href="favicon/favicon.ico?v=57" type="text/css">
<link href="js/css/dbwidgets.css" rel="stylesheet">
```

The Report Header, Report Body and Report Mouseover are used in On-form Reports. These styles (classes) must be identical apart from the following attributes which can be different:

- color
- background-color
- text-decoration
- cursor

Note that prior to Version 7 DesignBais hard coded some style attributes. If you are not using the standard DesignBais styles such as dbaisReportHeader and dbaisReportBody then you may need to add some attributes to your classes. An example is “cursor:pointer” which is now part of the DesignBais styles. You will also need to add a click event to your Report header style if you are not using the standard dbaisReportHeader.

Creating Multiple Styles within a single DesignBais Style definition

This technique allows multiple styles to be created in the one DBISTYLE record. It is possible to create a separate style record for “input.dbaisMVCross:disabled” but this method is recommended so as to keep related styles in the one place.

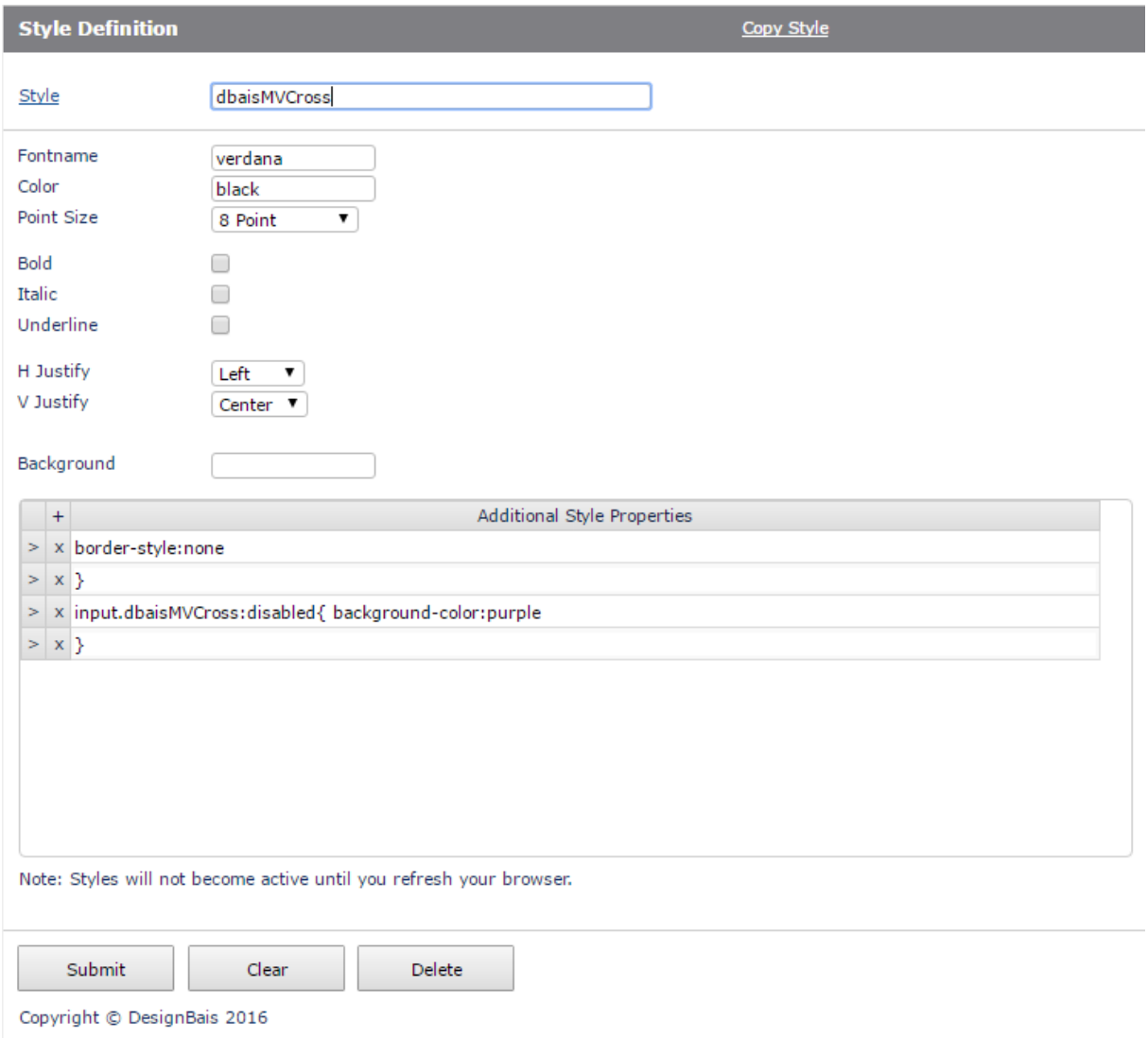
Normally, dbaisMVCross would be inserted into the DesignBaisStyle.css with a starting and closing brace as:

```
.dbaisMVCross{font-family: verdana;font-size: 8pt;color: black;vertical-
align:middle;border-style:none;}
```

To add an extra style we need the closing brace “}” to end the base style followed by a new style name with an opening brace “{“ to start the next style. This can be followed by any number of additional styles. The closing brace at the end is optional as it will be added by the program in the normal manner. In this case the extra style displays disabled fields as purple.

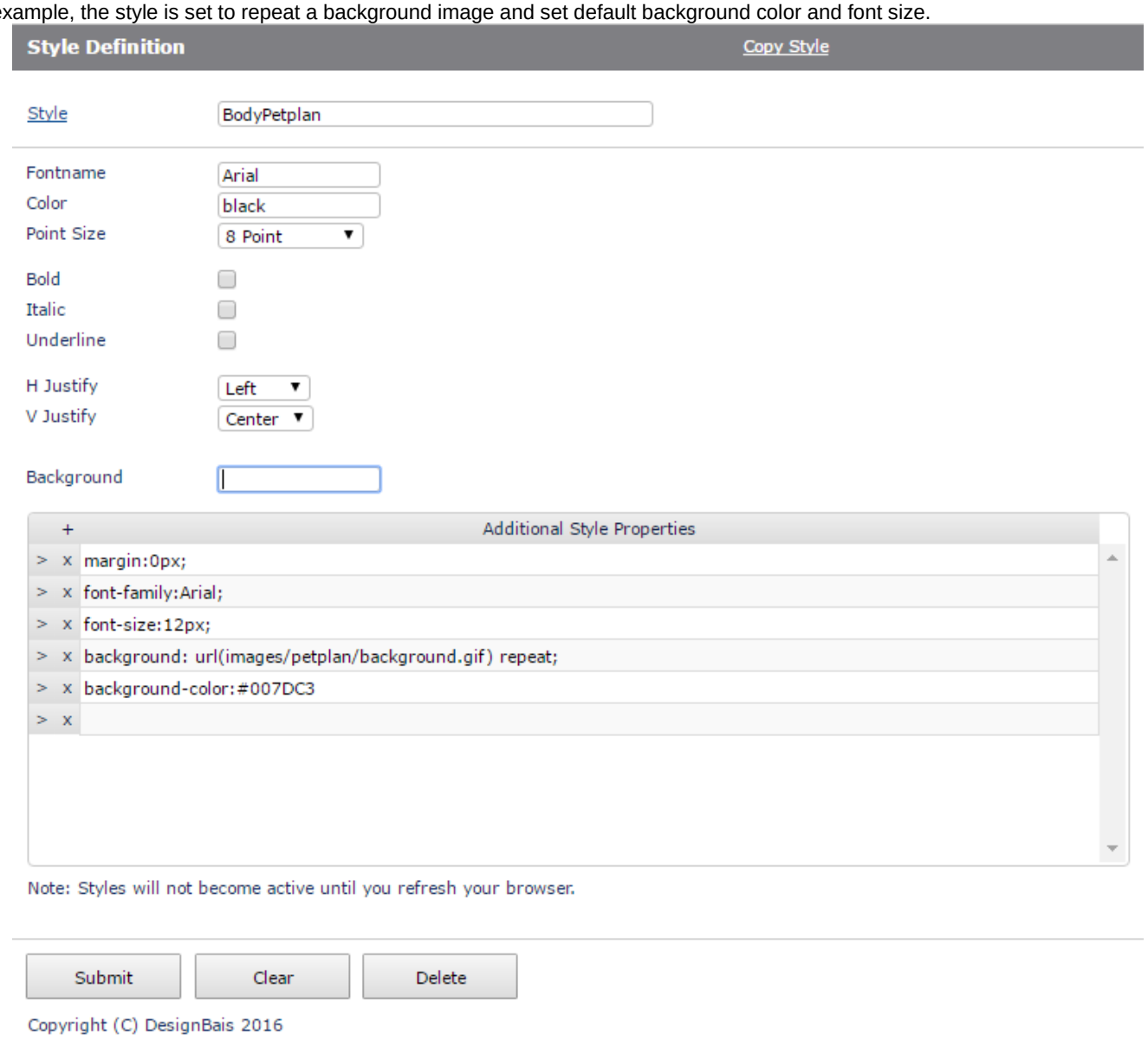
Note that semi-colons (;) are optional at the ends of each additional style multivalue.

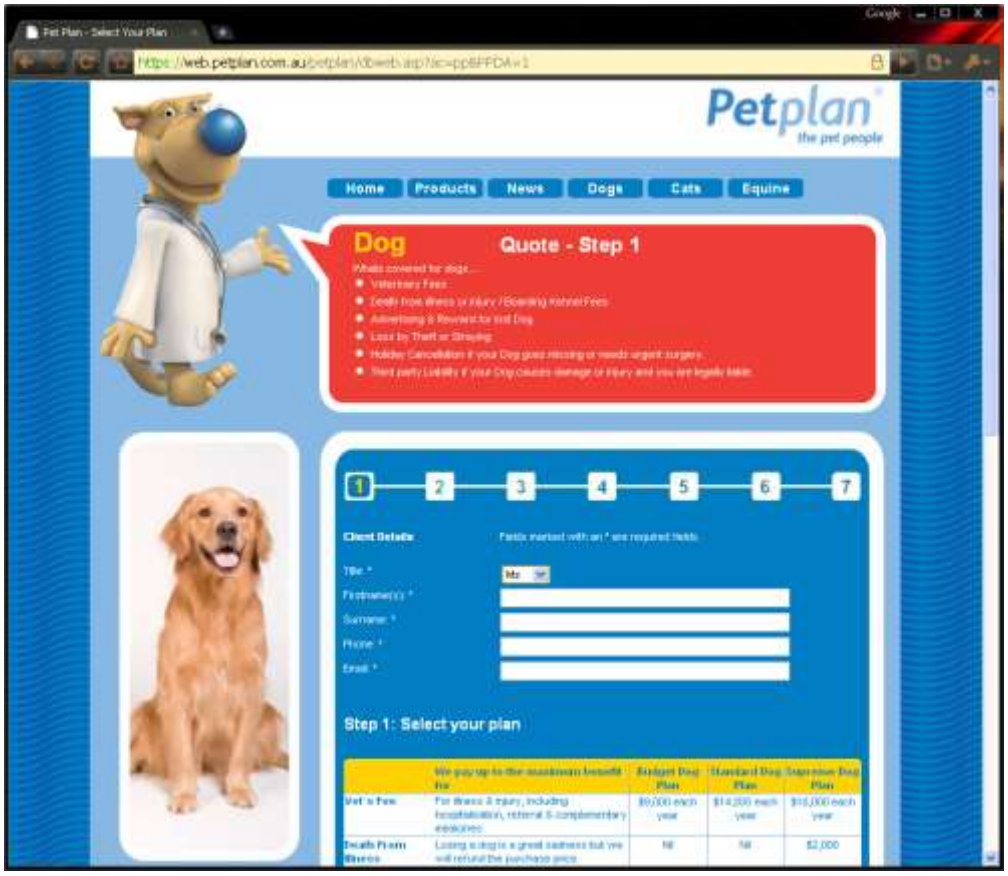
We end up with 2 style items:
.dbaisMVCross{font-family: verdana;font-size: 8pt;color: black;vertical-align:middle;border-style:none;}input.dbaisMVCross:disabled{ background-color:purple;}



Examples of Style Usage

In this example, the style is set to repeat a background image and set default background color and font size.





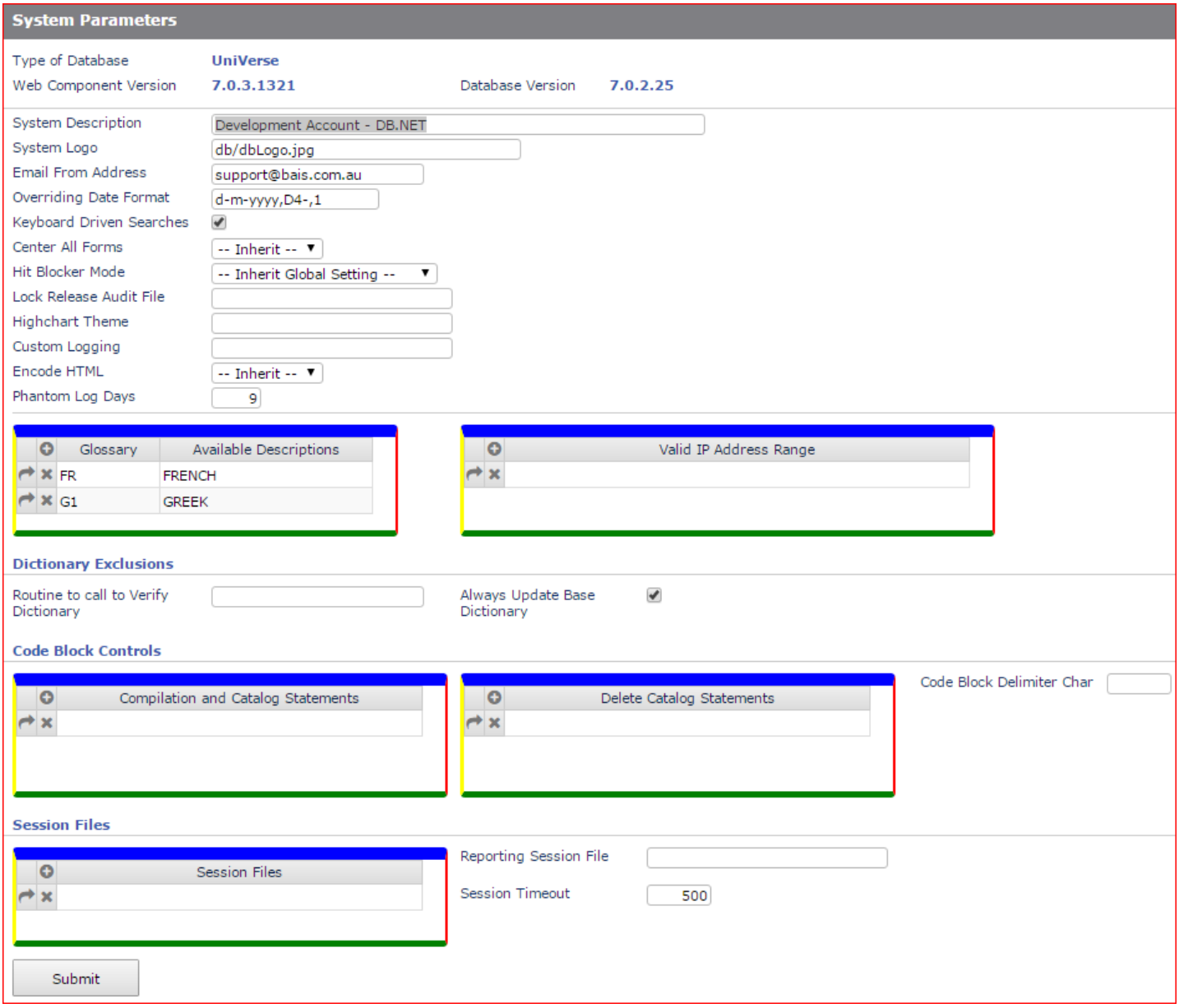
In this example, the MV Border Class contains Additional Style Properties defining border width and color.

Style Definition

Style Definition

Style	dbaisMVBorderClass
Fontname	verdana
Color	black
Point Size	8 Point
Bold	
Italic	
Underline	
H Justify	Left
V Justify	Center
Background	

Additional Style Properties
background-color:transparent
border-style:solid
-moz-border-radius: 4px
border-right:2px solid #ff0000
border-left:3px solid yellow
border-bottom:5px solid green
border-top:10px solid blue
-webkit-border-radius: 4px
border-radius: 4px

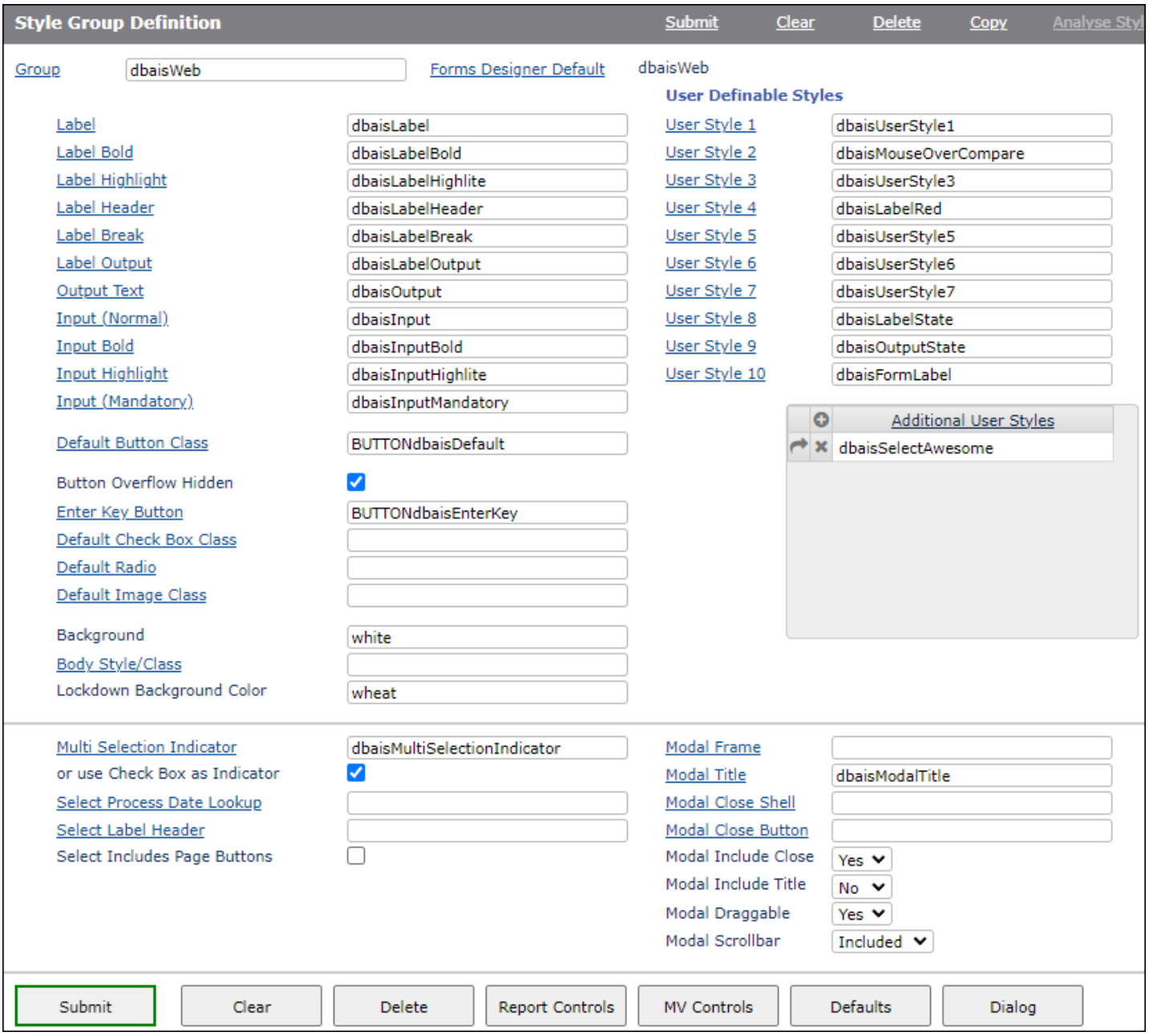

System Parameters
Type of Database
UniVerse
Web Component Version
7.0.3.1321
Database Version
7.0.2.25
System Description
Development Account - DB.NET
System Logo
db/dbLogo.jpg
Email From Address
support@bais.com.au
Overriding Date Format
d-m-yyyy,D4-,1
Keyboard Driven Searches
Center All Forms
-- Inherit --
Hit Blocker Mode
-- Inherit Global Setting --
Lock Release Audit File
Highchart Theme
Custom Logging
Encode HTML
-- Inherit --
Phantom Log Days
9
Glossary
Available Descriptions
FR
FRENCH
G1
GREEK
Valid IP Address Range
Dictionary Exclusions
Routine to call to Verify Dictionary
Always Update Base Dictionary
Code Block Controls
Compilation and Catalog Statements
Delete Catalog Statements
Code Block Delimiter Char
Session Files
Session Files
Reporting Session File
Session Timeout
500
Submit

Style Groups

Style groups provide the developer with a standard look and feel during the forms and report design processes. They also provide a standard look and feel for end-users. With applications being developed by multiple individuals with varying preferences for look of an application, it is important to control this variance by the application of a standard style (style group). The Style group labels various styles into a collection. The Style Group is given a name and then forms or reports are assigned a Style Group.

DesignBais tools use the 'dbaisWeb' and 'dbaisRep' Style Groups.

Changing the look of an entire application can be achieved by changing the attributes of the Style Group.



Group	The name to be assigned to the collection of styles (Style Group)
Forms Designer Default	Click to recall the style group that is flagged as the Designer Default Style Group in Forms Designer Defaults.
Analyse Styles	A tool to allow the developer to compare style groups. This function allows style groups to be compared and styles can be updated to match the styles in a reference style group.

Label Styles

These styles control the look of text and output labels placed on a DesignBais form or a DesignBais report.

Label	Assigns a Style to a standard label. This applies to any text-label added to a form or report.
Label Bold	Assigns a Style for a bold label. This applies to any text-label added to a form or report.
Label Highlight	Assigns a Style for a highlighted label. This applies to any text-label added to a form or report.
Label Header	Assigns a Style for a header label. This applies to any text-label added to a form or report.
Label Break	Assigns a Style for a break-style label. This usually is used for lines that separate sections on a form.
Label Output	Assigns a Style for Output Labels. These usually contain data from fields in a non-input mode.

Input Styles

These styles control the look of input fields when placed on a DesignBais form.

Input (Normal)	Assigns a style to a standard input field.
Input Bold	Assigns a style to bolded input fields.
Input Highlight	Assigns a style to highlighted input fields.
Input (Mandatory)	Assigns a style to mandatory input fields.
Default Button Class	Defines the standard Button Class (Display Style) for buttons that are placed on a form. Leaving this field blank will use the default button style (which is recommended).
Button Overflow Hidden	Buttons in Version 8 include overflow:hidden by default to prevent the button text from displaying outside the button boundaries. Set "Button Overflow Hidden" to off to get the Version 6 behaviour.
Enter Key Button	Defines the default Button Class (Display Style) for buttons linked to the Enter Key on a form. This class is applied to form buttons with no Display Class defined. If this field is left blank than the default button style will be applied.
Default Check Box Class	Assigns a style to check box fields.
Default Radio	Assigns a style to radio button fields.
Default Image Class	Assigns a style to image fields.
Background	Defines the standard background color for all forms created with the style group.
Body Style/Class	Defines the default style to be used for the body of the form. This is typically the space around the main form definition.
Lockdown Background Color	Defines the background color used to signify that Lockdown Mode is turned on.

Miscellaneous

Multi Selection Indicator During multi-selections, a color will be displayed in the selected rows to indicate a selection "on" state. The color of this is determined by the style assigned to this field.

or use Check Box When this is checked, multiple selections will be displayed with a check box instead of a highlighted field.

Select Process Date Lookup
This style is applied to any date hyperlinks in a Selection Process. If empty then dbaisSearchLabel is applied.

Select Label Header This style is applied to label headers in a Selection Process. If left null then the Label Header style is applied.

Select Includes Page Buttons
If not checked then the selection will use the newer compressed paging control. Check this field if you want the older paging control buttons applied.

Modal Styles

Modal Frame This style will be the default class for a div that holds a modal form.

Modal Title This style will be the default class for the title of a modal form.

Modal Close Shell This style will be applied to the modal form close button.

Modal Close Button This style will be applied to the modal form close button background shell.

Modal Include Close This option allows for the suppression of the close button. It is included by default.

Modal Include Title This option allows for the suppression of the form description in the modal form title. It is included by default.

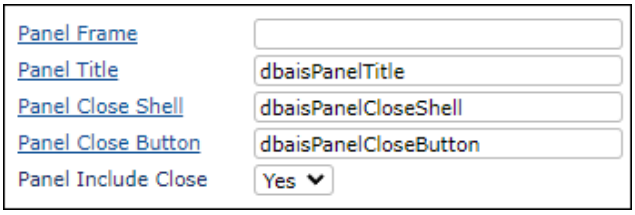
Modal Draggable This option allows for the suppression of the form draggable handle and the modal form title. The title is used as the draggable handle and hence both are suppressed by this option. They are included by default.

Modal Scrollbar This option allows for the suppression of the scroll bar in the modal form width. It is included by default.

User Definable Styles

User Style n There are ten additional user styles available. If they are defined for a Style Group, they will then be added to every Display Class field within the DesignBais Forms Designer and Report Designer properties screens. Following Release 8.3.3.14 an additional set of user styles can be defined using the multivalue field titled "*Additional User Styles*".

Panel Styles



Panel Frame The default class for the div that holds the form panel.

Panel Title The form panel title bar is used as the draggable handle.

Panel Close Shell This style will be applied to the form panel close button background shell.

Panel Close Button This style will be applied to the form panel close button.

Panel Include Close This option allows for the suppression of the close button in the form draggable panel. It is included by default but developers can suppress it in favour of a bespoke styled button.

Buttons

Report Controls Opens the Style Group Report Controls form

MV Controls Displays a sub form displaying Multivalue Controls for the Style Group

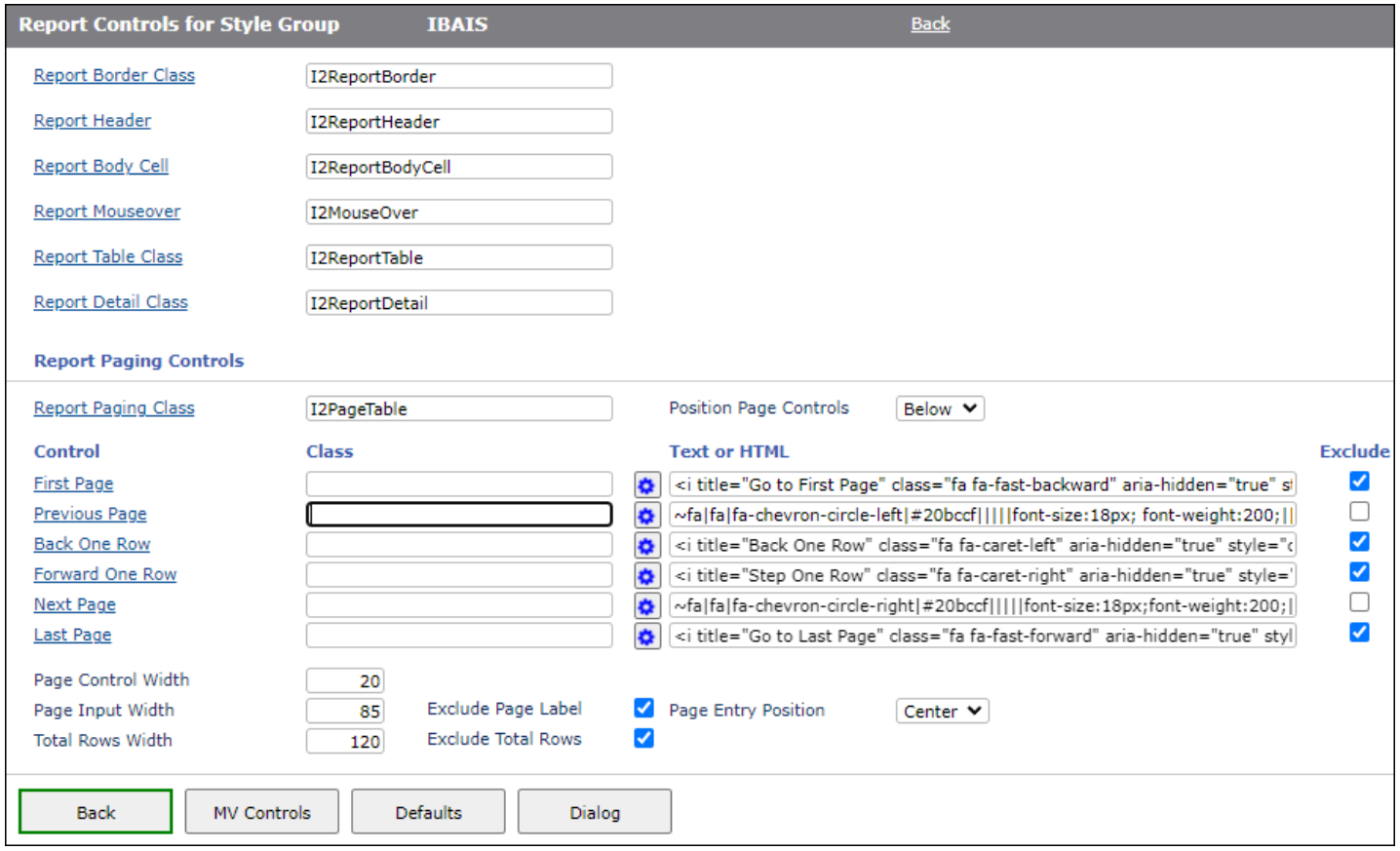
Defaults Opens the Style Group Forms Designer Defaults form

Dialog Opens the Dialog Box Controls for Style Group form

Report Controls for Style Group

These styles control the standard look of On-form Reports. They can be overridden by modifications to the OUTPUT.ATTR variable during On-form Report production.

Please refer to the On-form Reports section in the Common variable usage and subroutine interaction chapter.



Report Border Class Defines the style to be used for On-form Report borders.

Report Header Defines the style to be used for On-form Report headers.

Report Body Cell Defines the style to be used for the On-form Report body.

Report Mouseover Defines the style to be used for the On-form Report mouseover.

Report Table Class Defines the style to be used for the On-form Report row highlight on mouse hover.

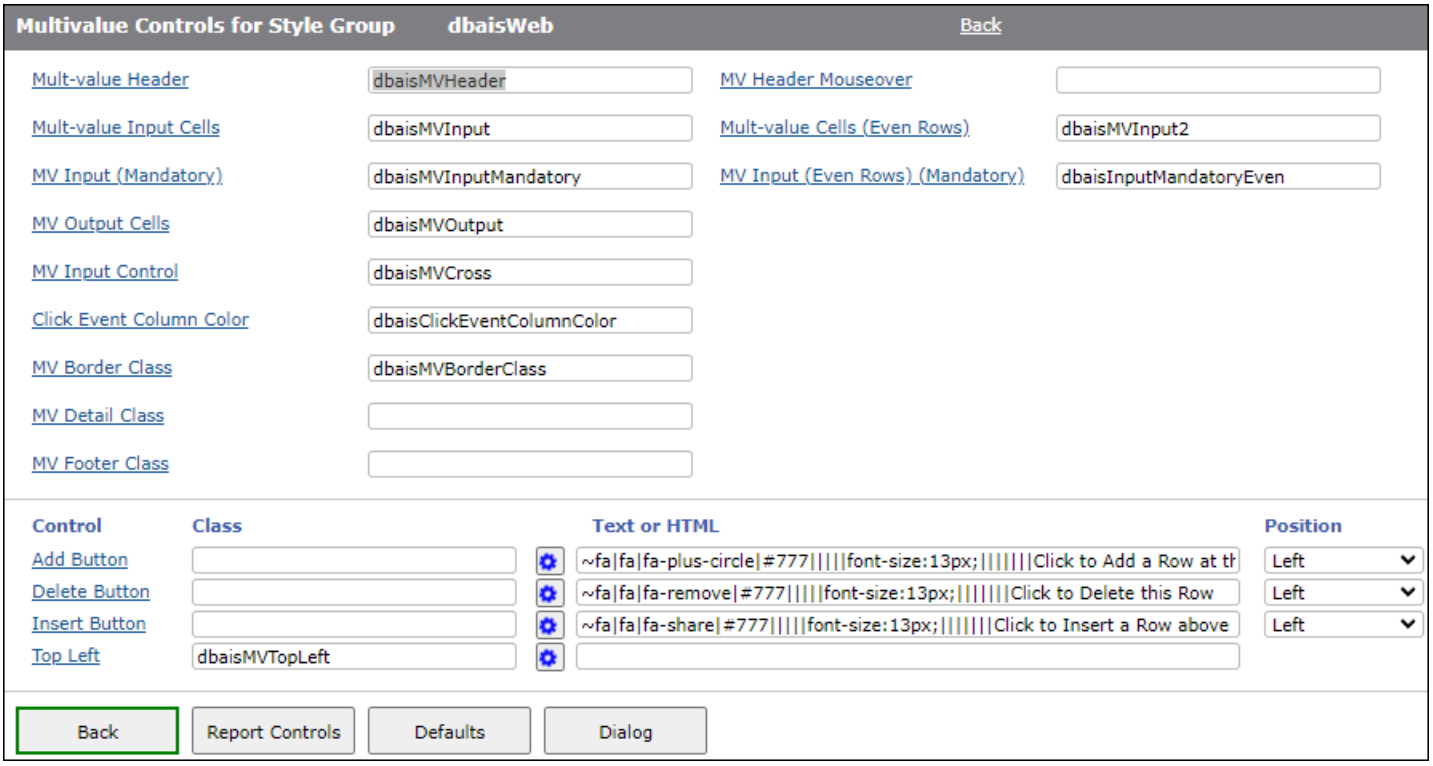
Report Detail Class Defines the style to be used for On-Form Report detail <divvy>'s. The background color can be set or a custom scrollbar for the detail can be configured.

Report Paging Class	This style is applied to the OFR Paging table added at the foot of an OFR with paging turned on. If empty then the MV Header style is applied.
Position Page Controls	The paging controls are positioned by default within the OFR depth as defined in the Forms Designer. This option allows the paging controls to be positioned below the OFR leaving the original report depth untouched.
First Page	This style is applied to the OFR Paging table First Page icon at the foot of an OFR with paging turned on. If empty then only the icon string is applied. If both are empty then the original default icon string will be applied.
Cog Button	Click to open the Field Font Awesome Icon Configuration form. [Applies to all the five paging options below.]
Text or HTML	DesignBais will place the awesome font fa-fast-backward to represent go to the first page. This text field allows you to replace the default icon with your own text which may be HTML. An entry in the First Page Class will override this text. [Applies to all the five paging options below.]
Exclude	Set to exclude the first page button from the OFR paging controls.
Previous Page	This style is applied to the OFR Paging table Previous Page icon at the foot of an OFR with paging turned on. If empty then only the icon string is applied. If both are empty then the original default icon string will be applied.
Exclude	Set to exclude the previous page button from the OFR paging controls.
Back One Row	This style is applied to the OFR Paging table Previous Row icon at the foot of an OFR with paging turned on. If empty then only the icon string is applied. If both are empty then the original default icon string will be applied.
Exclude	Set to exclude the back one row button from the OFR paging controls.
Forward One Row	This style is applied to the OFR Paging table Forward One Row icon at the foot of an OFR with paging turned on. If empty then only the icon string is applied. If both are empty then the original default icon string will be applied.
Exclude	Set to exclude the forward one row button from the OFR paging controls.
Next Page	This style is applied to the OFR Paging table Next Page icon at the foot of an OFR with paging turned on. If empty then only the icon string is applied. If both are empty then the original default icon string will be applied.
Exclude	Set to exclude the next page button from the OFR paging controls.
Last Page	This style is applied to the OFR Paging table Last Page icon at the foot of an OFR with paging turned on. If empty then only the icon string is applied. If both are empty then the original default icon string will be applied.
Exclude	Set to exclude the last page button from the OFR paging controls.
Page Control Width	The width to allow for each paging control. DesignBais uses this setting when adding a report to a form to check that the OFR is wide enough to hold the Paging Controls.
Page Input Width	The width to allow for the input of the page number to go to. Used when adding a report to a form to check that the OFR is wide enough to hold the Paging Controls.
Total Rows Width	The width to allow for the Total Rows text. Used when adding a report to a form to check that the OFR is wide enough to hold the Paging Controls.
Exclude Page Label	Set to exclude the page label from the OFR Paging Controls.

Exclude Total Rows	Set to exclude the Total Rows nn message from the OFR Paging Controls.
Page Entry Position	The go to page entry is positioned to the right of the Paging Controls by default. This option allows the go to page entry to be positioned to the left or in the center of the controls.

MV Controls button

Styles relating to multivalue grid controls are grouped together on this form.



These styles control the look of the MultiValue grid controls that are placed on a DesignBais form.

Multi-Value Header	Defines the style to be used for MultiValue headers.
MV Header Mouseover	Defines the style to be used for MultiValue header mouseover.
Multi-Value Input Cells	Defines the style to be assigned for odd rows in a MultiValue control.
Multi-Value Cells (Even Rows)	Defines the style to be assigned for even rows in a MultiValue control. Typically this would be the same as the odd-row definition with a slight variation in background color.
MV Input (Mandatory)	Defines the style to be used for mandatory MultiValue fields.
MV Input (Even Rows) (Mandatory)	Defines the style to be assigned for mandatory even rows in a MultiValue control. Typically this would be the same as the odd-row definition with a slight variation in background color.
MV Output Cells	Defines the style to be used for MultiValue output fields.
MV Input Control	Defines the MultiValue Input control class.
Click Event Column Color	Defines the style to be used when a click event is placed on a MultiValue field.
MV Border Class	Defines the MultiValue grid border style. You can style the border width in the “MV Border Class”, however you will lose space inside the grid if you do. Make the grid fields wider, if necessary, to allow for this.

MV Top Left Defines the MultiValue grid top left corner style.

MV Add Button Defines the MultiValue grid button to add a row.

MV Delete Button Defines the MultiValue grid button to delete a row.

MV Insert Button Defines the MultiValue grid button to insert a row.

Typically, DesignBais will place a + sign, X character and > symbol to represent add, delete and insert. If you wish to use your own styling you can enter the style/class names within these three fields. If a style name is used, DesignBais does not include the characters as it expects the styling to include the representative of these functions.

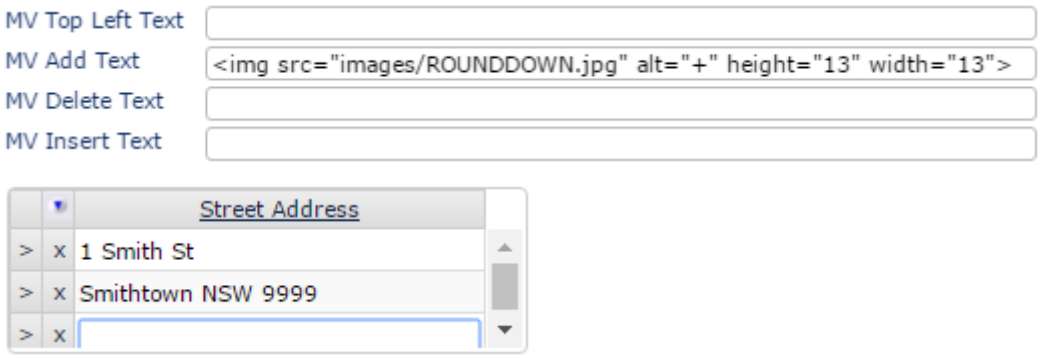
MV Top Left Text The text fields allow you to replace the DesignBais default " " with your own text which may be HTML. An entry in the MV Top Left Class will override this text.

MV Add Text This text field allows you to replace the + sign with your own text which may be HTML. An entry in the MV Add Button Class will override this text.

MV Delete Text This text field allows you to replace the x character with your own text which may be HTML. An entry in the MV Delete Button Class will override this text.

MV Insert Text This text field allows you to replace the > symbol with your own text which may be HTML. An entry in the MV Insert Button Class will override this text.

This example demonstrates how to apply a different image for the “add a row” button:



If the website db.config file has enabled the cross-site scripting (XSS) shield:

<enableXSSshield>true</enableXSSshield>

then you will not be able to save the HTML code in these four fields.

Forms Designer Defaults

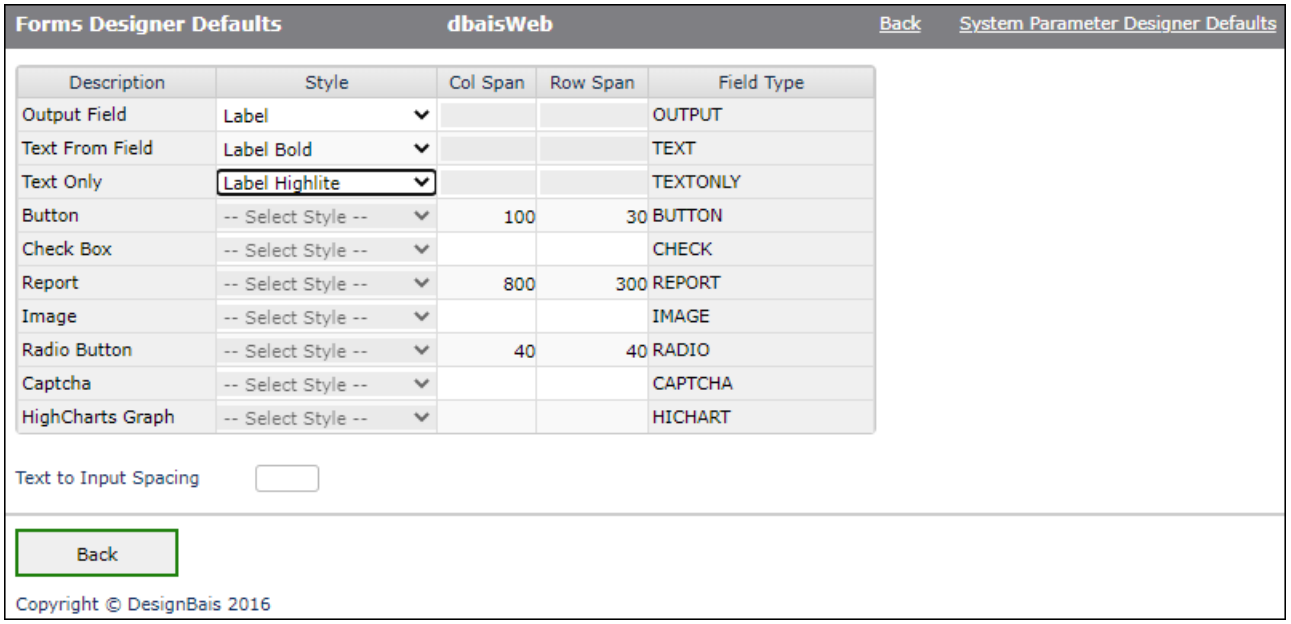
Forms Designer Defaults

Use this form to enter default values for use in Forms Designer for forms using the style group.

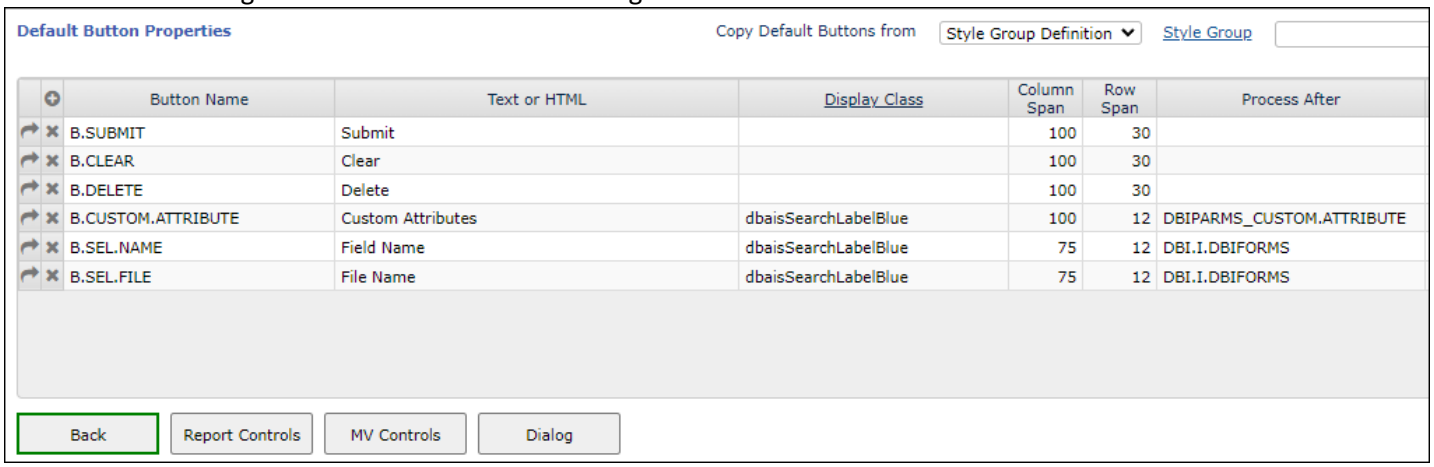
From Release 8.3.3.6 the Forms Designer tool allows form elements to be sized based on the the styles defined in the style group designated for the form.

These default style, column span, row span and input spacing values are used when the form element style does not define a default value. If these default values are blank then Forms Designer uses the set of defaults defined in the System Parameters Designer Defaults.

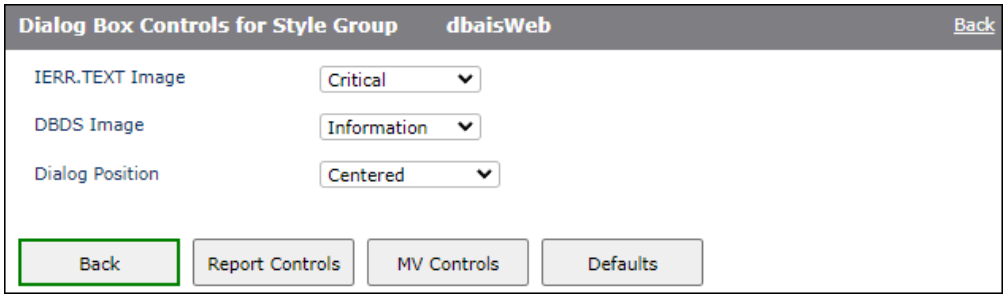
If no defaults are defined then the traditional defaults that were effectively hard-coded within the DesignBais engine are applied.



Use the fields below to define default button properties. These buttons will then be available to select when adding a button element in forms designer.



Click the System Parameter Designer Defaults link to display the System Parameters Designer Defaults in enquiry mode.

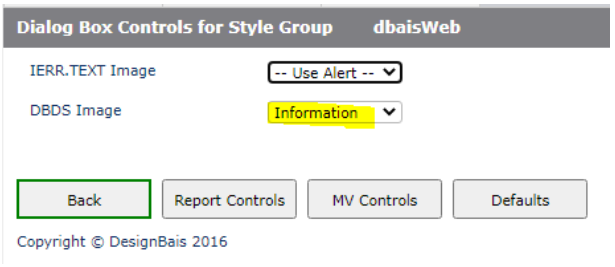


IERR.TEXT Image — This determines the image to be applied when IERR.TEXT is displayed in a standard Dialog Box. If not provided then the original alert display will be used.

DBDS Image — This determines the image to be applied when DBDS is displayed in a standard Dialog Box. If not provided then the original alert display will be used.

Dialog Position — Determines the position of the dialog box. By default the dialog is positioned under the field. Select Centered if dialogs are to be always positioned in the centre of the browser display area.

The Style Group definition now has an option to use a Dialog box rather than a simple alert box.



The Dialog box option will place the dialog near the field that triggered the display.

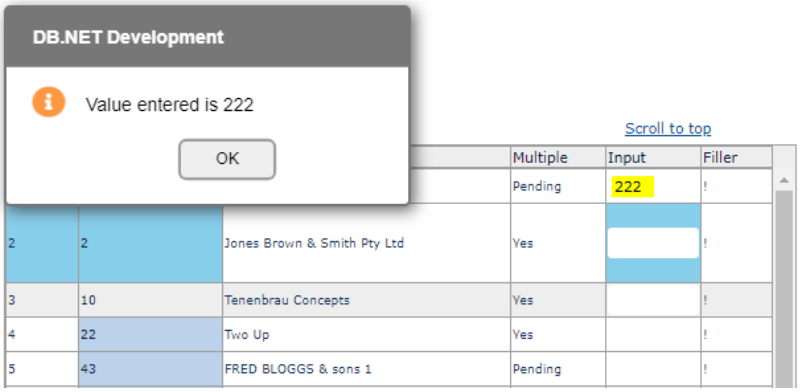
The drawback is that the Dialog invokes another database server hit after the OK button is clicked.

Developers should note that if the form element generating the dialog is the last field on the form (page) and the field value is changed and the user tabs out, then the focus moves to the first element on the page (i.e. the page scrolls). To prevent that, you can use the following custom attribute on that element:

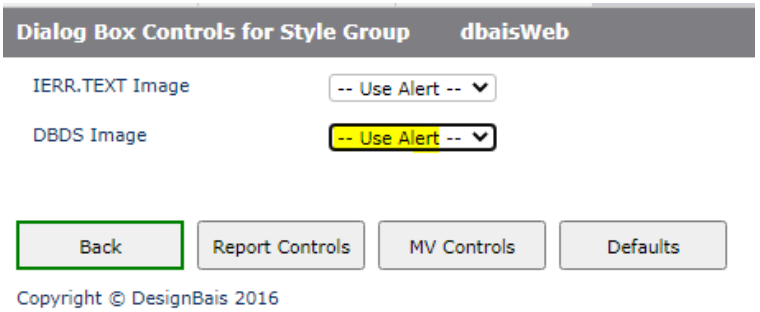
```
onkeydown="if (event.keyCode==9){event.preventDefault();event.stopPropagation();genericEventCall($(this).attr(\'id\'),\'change\')};"
```

This will cancel the tab-out event and fire change event for the field. Your validation code can then decide how to proceed, such as using DBRETURN.TO.FIELD to set focus, or display a dialog box message.

From an OFR cell the dialog is placed near the cell that triggers the message:

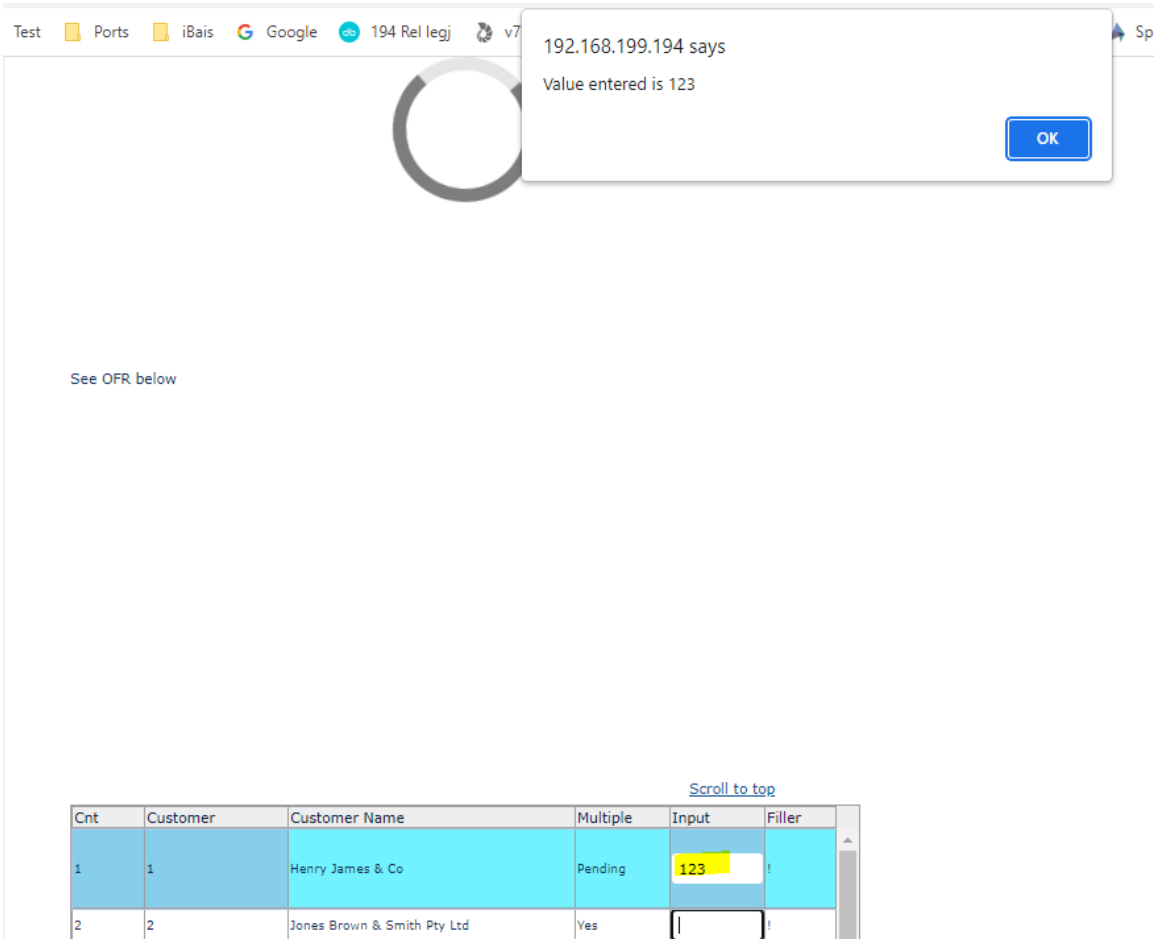


The traditional alert box setting displays centered at the top of the browser window.



Message from an input field:

Message generated by an on-form report cell:



The use of a dialog box, rather than a javascript Alert, to display IERR.TEXT prevents the developer from displaying a dialog box in the same event. DesignBais does not allow for two dialog boxes in the same event. The same applies to DBDS messages displayed using a dialog box.

Using Dialog Box for IERR.TEXT and DBDS from an On-form Report

The developer must specify where focus should return to after the user responds to the DBDS dialog box. This can be accomplished using DBRETURN.TO.FIELD.

In the case of IERR.TEXT the value in the field must be reverted so the developer needs to use DBREPORT.UPDATE.

Example:
In this example the data populating the OFR is held in DBOTHER.RECORD(12). Assume that a message is displayed when the value entered into the OFR input field in column 5 differs from the existing value in the cell.

```
            CASE COL = 5
               IF DBVALUE # DBOTHER.RECORD(12)<ROW> THEN
                  MSG = 'Value entered is ':DBVALUE
                  IF DBWORK<DBC.RADIO.WK> = "Y" THEN
```

- use IERR.TEXT to display the message

```
                     IERR.TEXT<-1> = MSG:' [IERR.TEXT]'
                     * this is considered an error so revert the value in the cell to DBOTHER.RECORD(12)<ROW>
                     DBREPORT.UPDATE<1,1> = "R.REPORT1"
                     DBREPORT.UPDATE<2,1> = ROW
                     DBREPORT.UPDATE<3,1> = COL
                     DBREPORT.UPDATE<4,1> = DBOTHER.RECORD(12)<ROW>
                     DBREPORT.UPDATE<6,1> = 'I'
                     IF DBREPORT.UPDATE<4,1> = '' THEN
                        DBREPORT.UPDATE<7,1> = 1
                     END
                  END ELSE
```

- use DBDS to display the message

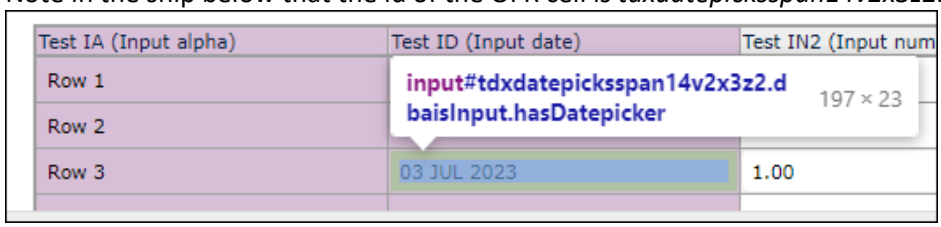
```
                     DBDS<-1> = MSG:' [DBDS]'
                     * using DBDS indicates that this is not an error but just a message so we direct focus to the
                     * following row
                     DBRETURN.TO.FIELD = "R.REPORT1~|":ROW+1:".":COL
                  END
               END
               DBOTHER.RECORD(12)<ROW> = DBVALUE
```

Note that for date input OFR cells (OUTPUT.TYPE(PRN)<1,n> = 'ID') the developer must append the string '.ID' to the field name specified in DBRETURN.TO.FIELD. This is required in order for the DesignBais engine to set the correct element id name e.g. *tdxdatepicksspan14v2x1z2*. In the example following the second column of the on-form report has a date input:

```
IF EVENTTYPE = click THEN
   IF COL = 2 THEN
      DBRETURN.TO.FIELD = "R.REPORT1~|":DBREPORT.CELL:".ID"
   END ELSE
      DBRETURN.TO.FIELD = "R.REPORT1~|":DBREPORT.CELL
   END
END
```

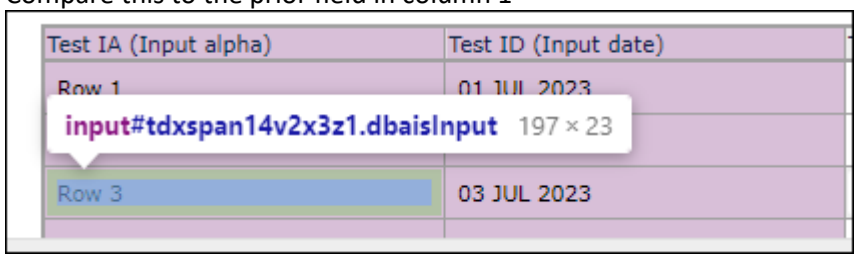
DBDS or IERR.TEXT when used with the Dialog Box display option can loop as you click the OK button, leaving the user focussed in the dialog box. This requires the developer to set DBRETURN.TO.FIELD after a click event.

Note in the snip below that the id of the OFR cell is *tdxdatepicksspan14v2x3z2.d*



- 14 is the xml field number from the form (including header fields)
- 2 is the form level DBWLEVEL (this form is 1 level up from the base form)
- 3 is the row
- 2 is the column

Compare this to the prior field in column 1



- 14 is the xml field number from the form (including header fields)
- 2 is the form level DBWLEVEL (this form is 1 level up from the base form)
- 3 is the row
- 1 is the column

DesignBais assigns the field id prefix *datepicks* for date input fields.

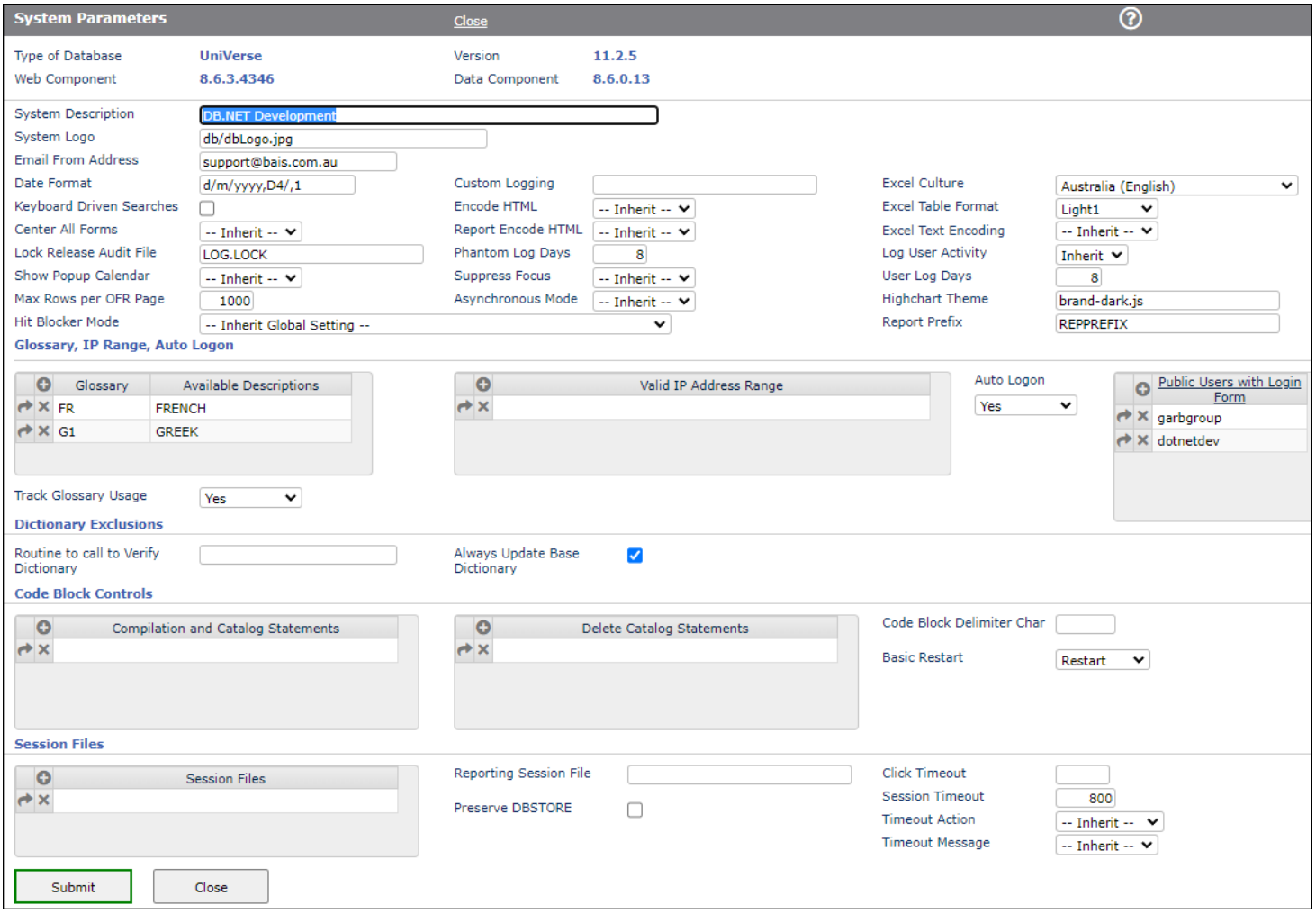
Refer to the documentation for Controlling Cursor Behaviour.

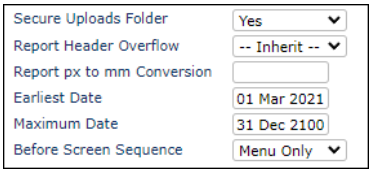
Chapter 11 – System Parameters

System Parameters

The System Parameters control various system-based functions at an account level. Commencing in Release 8.9.0.1 the System Parameters are selected from a new form with each parameter function selected from a side menu. Global Parameters can be accessed from this menu. There is a *Home* option to return you to the main tools form.

From 8.9.0.1 the parameter forms check for a lock set by another user accessing the function. You can choose to proceed in *Enquiry* mode.





Prompts

Type of Database Displays the Database type for this DesignBais server. Supported types are:

- UNIVERSE
- UNIDATA
- D3
- OAS
- JBASE
- OPENQM
- MVON4

Version The version of the database.

Web Component
Data Component Displays the current version of both Web and Data Component. To ensure that DesignBais versions maintain full functionality, the first two digits of each component must match.

Example. 8.3.1.6 Data Component will operate with 8.3.1.4017 of the Web Component.
8.2.1.7 Data Component will not operate with 8.1.0.1 of the Web Component.

System Description

Defines the description for the current DesignBais account or directory. This will be used as a title header in the browser form.

System Logo

Defines the logo (image) to be placed in the top left hand corner of the DesignBais window. This can be overridden by menu definitions.

Email From Address

This field is used to default the from email address in emails sent from DesignBais. The common variable DBEMAILFROM can be set programmatically to override this field.

Date Format

This field is used to ensure that date fields are interpreted correctly by the database server.

This must exist in either of the following two formats:

d-m-yyyy,D4,1 for International date formats
m-d-yyyy,D4,0 for USA date formats

For ease of maintenance you need only enter:

- m- this sets "-" as the separator, 0 for USA date format, 4-digit year
- m-yy this sets "-" as the separator, 0 for USA date format, 2-digit year
- d- this sets "-" as the separator, 1 for International date format, 4-digit year
- d-yy this sets "-" as the separator, 1 for International date format, 2-digit year

The number of y's sets the number of year digits and only 2 or 4 are valid. Any other will default to 4.

This field may be left blank in which case the date format may be defined by the format held in the DBIGLOBAL file, if one exists.

If DesignBais does not find a date format from System Parameters or DBIGLOBAL then it will default to the d/m/yyyy,D4,1 International date format.

User, System & Global "Date Format" may now contain "M" or "ML" in the month position to indicate that dates are to display the 3 character abbreviated upper (M) or lower (ML) case month name. This will set the output format to "D4" (or "D2") for "M" and "D4L" (or "D2L") for "ML". The space character will be then used as the date separator.

"ML" is not valid for UD or D3.

Date fields may require a dummy server hit (process after subroutine) in order to display the required format.

DesignBais System and Global Parameters exist for Earliest Date and Maximum Date. They are loaded into DBDATEFORMAT<1,4> and <1,5>. They will be used to improve the error message for date fields that do not have a field property date range setting viz: "A valid date is required between 01/03/2023 and 21/12/2500" assuming the earliest allowed date is 01/03/2023 and the maximum allowed date is 21/12/2500.

These parameters will prevent errors caused by the default database date conversion which currently only displays four digits for the year. An entry of "01/01/52023" will display as "01/01/2023" and a user will not be aware that an error has been made.

KeyBoard Driven Searches

This entails the addition of a row number field and row number column to each search form.

When enabled all search forms will have a Select Row field added automatically. They will also have a row number added as the first column on every search form. The row number is the number corresponding to the actual row on each page. It is not a record number.

As the Select Row field always has focus if there is an active page, the end-user can simply enter the row number followed by the Tab or Enter Key.

If the selection has multiple pages, the end-user may use the Page Up and Page Down keys (on the keyboard) to page through the selected records.

******* When Keyboard driven searches are turned on, the DBFORMLOCAL variable is being used. In order to ensure that you always have the latest search form when testing or developing a selection process, it is advisable to refresh your browser regularly. When the selection process is live, this is not necessary ***

Center All Forms

Forms with Form Centered set to Inherit will use this System Parameter setting. If "Inherit" is selected for this System Parameter then the Global Setting will be applied. Select "Yes" to indicate that all forms are to be centered by default. "No" will Left Align forms with "Inherit".

Lock Release Audit File

When a lock is released (Exclusive Locks) by default, there is no audit kept for the released record. If there is a file name entered in this field, an audit will be kept if the named file can be opened. The audit file created has the following format.
Attr<0> - Id of the release lock
Attr<1> - Released by user
Attr<2> - Date of the release
Attr<3> - Time of the release
Attr<4> - Originally locked by User
Attr<5> - Session Id of the original lock
Attr<6> - Path to the original lock
Attr<7> - Record Id

Show Popup Calendar

This feature enables the display of the Date Picker Calendar. The default behaviour is to not display the calendar. Selecting On Focus causes the Date Picker to display on any date field as it gains focus. Note that when the default setting is used the Date Picker can still be displayed using a button. Refer to the subroutine **DBI.G.CALENDAR** in this Manual. The global parameter setting, if present, overrides this setting.

If the date field is the first input field on a form and the popup calendar does not display below the date field then you may have to set the tab index settings so that buttons are not indexed before the date field.

Max Rows per OFR Page

Paging controls will be added to any OnForm Report with more rows than this maximum per page. Each page of the OFR will contain this number of rows or less. This maximum value can be overridden by the developer in basic code by setting a numeric value in PROCESS.REFRESH<2>.

Hit Blocker Mode

This feature enables the developer to control the action of subsequent events following a button event, input event, and a range of other events such as Image or Report.

Hit Blocker Mode may also be set on Global Parameters or applied at Field Level. The Global setting will be applied if both Systems Parameters and Field Hit Blocker Mode are set to ‘Inherit’.

Values are:
‘’ = Inherit Global Setting
0 = Element not disabled. Subsequent events are queued and sent in correct order.
1 = Element not disabled. Subsequent events are blocked. Events are lost and page is disabled.
2 = Element disabled but enabled on return. Subsequent events queued and sent in correct order.
3 = Element disabled but enabled on return. Subsequent events are blocked.
4 = Element disabled & kept disabled. Subsequent events queued and sent in correct order.
5 = Element disabled and kept disabled on return. Subsequent events are blocked.

Custom Logging

Custom Logging routine. This subroutine will be called with either LOG_START or LOG_STOP call argument in attribute 1. Custom parameters may be provided after a | (pipe) character e.g. UTL.LOGGING|0,500.

The custom logging routine will be called with the following attribute separated argument list:

<1> LOG_START or LOG_STOP

<2> Custom Parameters
<3> Date
<4> Time
<5> SYSTEM(12) for Universe (current system time in seconds or milliseconds)
<6> PROCESS.EVENT
<7> PROCESS.EVENTSOURCE
<8> PROCESS.PARAMETER
<9> SESSION.ID
<10> DBEX.SESSION.ID
<11> WEBLOGON
<12> DBIACCOUNT
<13> SCREENROOT
<14> Input Sring for LOG_START, Output String for LOG_STOP

Example of basic subroutine to implement logging:

```
SUBROUTINE DBI.G.LOG(LOGSCRAPE)
* Custom Logging Sample
*
* LOGSCRAPE    = OS.DATAIN<1> ;* Either LOG_START or LOG_STOP
* LOGSCRAPE<2> = FIELD(OS.DATAIN<2>,"|",2)
* LOGSCRAPE<3> = DATE()
* LOGSCRAPE<4> = TIME()
* LOGSCRAPE<5> = SYSTEM(12)
* LOGSCRAPE<6> = PROCESS.EVENT
* LOGSCRAPE<7> = PROCESS.EVENTSOURCE
* LOGSCRAPE<8> = PROCESS.PARAMETER
* LOGSCRAPE<9> = SESSION.ID
* LOGSCRAPE<10> = DBEX.SESSION.ID
* LOGSCRAPE<11> = WEBLOGON
* LOGSCRAPE<12> = DBIACCOUNT
* LOGSCRAPE<13> = SCREENROOT
* LOGSCRAPE<14> = OS.DATAIN<3> ;* Will be Input string for LOG_START and Output String for
LOG_STOP
* LOGSCRAPE<15> = IERR.TEXT    ;* For LOG_ERROR
*
$INCLUDE DBINET DBI.COMMON
$INCLUDE DBINET E.DBISESSIONS
*
   OPEN '','DBIXMLLOG' TO F.DBIXMLLOG ELSE RETURN
   *
   XX = 20
   LOGSCRAPE<XX> = 'DBTHISSTEP=':SESSION.REC<DBISS.THISLOAD>   ; XX+=1
   LOGSCRAPE<XX> = 'APP.LOGIN =':SESSION.REC<DBISS.LOGIN.ID>   ; XX+=1
   LOGSCRAPE<XX> = '@USERNO =':@USERNO   ; XX+=1
   *
   TYPE = ''
   BEGIN CASE
      CASE LOGSCRAPE<1> = 'LOG_START'
         TYPE = "I"
      CASE LOGSCRAPE<1> = 'LOG_STOP'
         TYPE = "O"
      CASE LOGSCRAPE<1> = 'LOG_WS_START'
         TYPE = "WI"
      CASE LOGSCRAPE<1> = 'LOG_WS_STOP'
         TYPE = "WO"
      CASE LOGSCRAPE<1> = 'LOG_ERROR'
         TYPE = "E"
      CASE LOGSCRAPE<1> = 'LOG_REP_START'
         TYPE = "RI"
         * LOGSCRAPE<XX> = "DBUSAGEVAR:" ; XX += 1
         * LOGSCRAPE<XX> = DBUSAGEVAR    ; XX = DCOUNT(LOGSCRAPE,AM) + 1
         LOGSCRAPE<XX> = "DBPRINTSELECT:" ; XX += 1
         LOGSCRAPE<XX> = DBPRINTSELECT    ; XX = DCOUNT(LOGSCRAPE,AM) + 1

      CASE LOGSCRAPE<1> = 'LOG_REP_STOP'
         TYPE = "RO"
         * LOGSCRAPE<XX> = "DBUSAGEVAR:" ; XX += 1
         * LOGSCRAPE<XX> = DBUSAGEVAR    ; XX = DCOUNT(LOGSCRAPE,AM) + 1
```

```
            LOGSCRAPE<XX> = "DBPRINTSELECT:" ; XX += 1
            LOGSCRAPE<XX> = DBPRINTSELECT    ; XX = DCOUNT(LOGSCRAPE,AM) + 1
            LOGSCRAPE<XX> = "DBSENTENCE:" ; XX += 1
            LOGSCRAPE<XX> = @SENTENCE    ; XX = DCOUNT(LOGSCRAPE,AM) + 1

      END CASE
      IF TYPE = '' THEN RETURN
      *
      READU CNT FROM F.DBIXMLLOG,"LOG.CNT" ELSE CNT = 0
      CNT += 1
      WRITE CNT ON F.DBIXMLLOG,"LOG.CNT"
      *
      ID = CNT:"*LOG*":TYPE
      WRITE LOGSCRAPE ON F.DBIXMLLOG,ID
      RETURN
END
```

Encode HTML

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user.

XSS vulnerabilities may occur if either input coming into web applications is not validated or output to the browser is not HTML encoded.

The Encode HTML feature is provided to prevent XSS attacks. At the System Parameters level the setting can be:

Inherit	-	Inherit the setting from the Global Parameters
No	-	Do not encode HTML meaning that XSS attack is possible
Yes	-	Encode HTML to prevent XSS attack

Report Encode HTML

This setting applies to reports built in the Report Designer only. Use the Encode HTML for Form elements.

Output fields containing HTML elements should be encoded to prevent XSS injection attacks. Encoding HTML means converting characters that have meaning in HTML to their display only string e.g. < is encoded as & l t ; (without the spaces).

This means that any HTML will display as entered but will not be active in the browser.

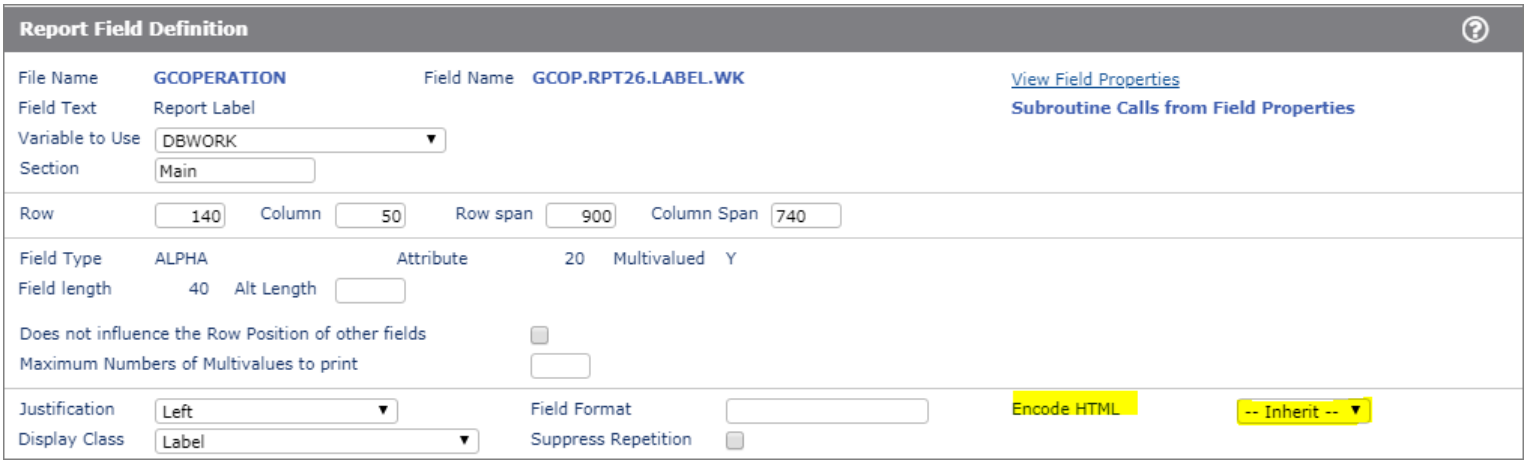
Encoding may be set to Inherit, Yes or No.

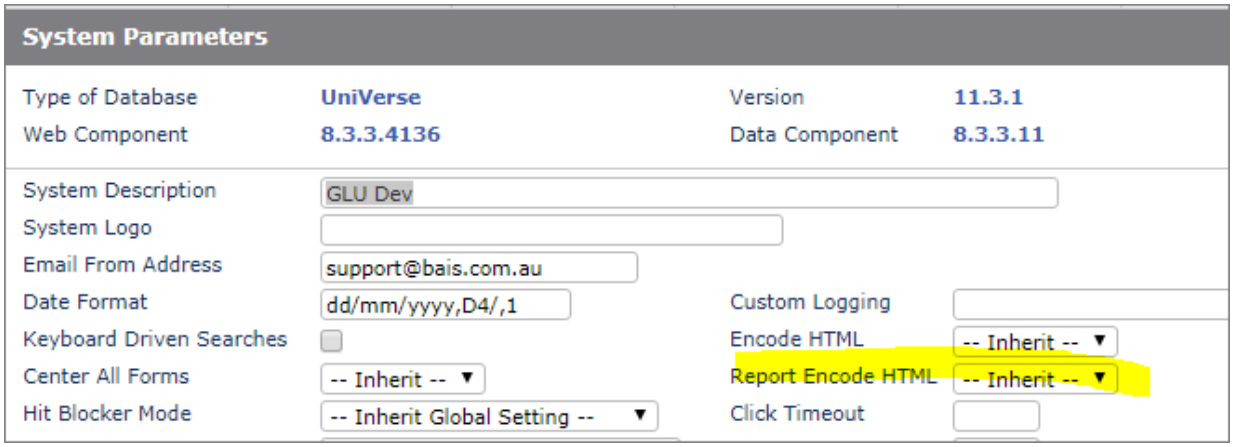
Encoding may be set on Output Fields. If an Output Field is set to Inherit (the default action) then the System Parameter setting will be applied, if the System Parameter value is Inherit then the Global Setting will be applied.

The Global Setting will default to Yes (HTML encoding) to provide backwards compatibility for existing applications.

It is recommended that HTML Encoding be turned on with only fields requiring active HTML elements having No encoding.

The following example will assist the developer to set up HTML Encoding for DesignBais reports. Report HTML encoding was introduced to provide a way to stop white-space wrapping causing overlapping text. This report field will inherit the Encode Html setting from the System Parameters setting, or if this is set to *Inherit*, from Global Parameters:





This will allow HTML to be embedded in reports.

A report that includes HTML code, for example:

```
DBWORK<GCOP.RPT.LABEL.WK, 26, SVM.CTR> = '<b>Case No:
</b>':PROF.ID  :STR(' ',5):'<b>Police Ref: </b>':PROFILE.REC<GCCA.POLICE.REF>
```

will now display correctly:

Phantom Log Days

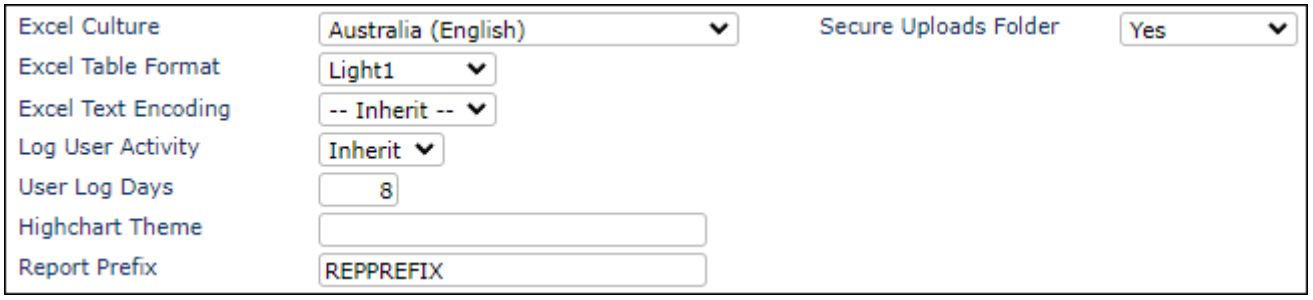
Report and other Phantoms are logged on the DBISTATS file in order to track their progress. Records will be purged after the number of days entered.

Suppress Focus

DesignBais by default controls field focus by sending a javascript focus() command after a change event triggered when the user tabs out of a field or presses the enter key in a field. This requires strict control of the sequencing of events between the web component and the database component which may interfere with users typing ahead. Selecting Inherit allows for a Global setting, Yes will suppress the sending of focus() commands in this System, while No is the default behaviour. Any DBRETURN.TO.FIELD will take precedence.

Asynchronous Mode

By default DesignBais runs with asyncronous ajax calls since synchronous mode has been deprecated by browser vendors. Developers may, however, choose to run in synchronous mode by selecting *No* in this field. This setting may be inherited from the value in the Global Login Parameters. Hit Blocker modes will be ignored if synchronous mode have been selected. Note that a DesignBais application that relies on synchronous ajax calls may cease to function correctly once the support for synchronous ajax calls is no longer available in the web browser being used.



Excel Culture

The Excel Culture is used by the conversion to Excel component to determine if an exported value is a date. The output format is determined by Date Format parameter.The user setting if present, overrides the System Parameter which in turn, overrides the Global setting.

Excel Table Format

This setting determines if the data exported to Excel is to be displayed in table format and if so, the style of table. If Inherit is chosen then the setting will be Inherited from the Global setting. If No Table Format is selected then this will mean that the raw data is exported to Excel with no table formatting.

Excel Text Encoding

This setting determines if the data exported to Excel is to be encoded as ASCII (DEFAULT) or as utf-8. Using UTF8 together with the Culture setting allows Currency symbols to be applied when the Excel file is produced. May be overridden by the User setting. If not entered on the user and not entered here or Inherit is chosen then the setting will be Inherited from the Global setting.

Log User Activity

This User Activity may be logged on the file DBIUSERLOG and reported. This may be set at System or User level as well. Users with Log User Activity set to "Inherit" with use the System Parameters value. Systems with Log User Activity set to "Inherit" will use the Global Parameter. Select "No" to stop logging user activity in the DBIUSERLOG file. "Yes" is the default. Stored in the DBIGLOBAL record USERLOG.

User Log Days

The number of days to keep DBIUSERLOG records. See Log User Activity above.

Highchart Theme

The name of the Highchart Theme to be used in this application. The name entered here will override the Global Highchart Theme Parameter. If left blank then the default Highchart Theme is applied. The available theme javascript files are held in the DesignBais website folder charts/themes.

Developers can also use OUTPUT.AT<50> to set Highchart parameters:

```
OUTPUT.AT<50> = 0                          ;* show export menu button (1=hide)
OUTPUT.AT<50,2> = "images/yourImage"       ;* export menu image
OUTPUT.AT<50,3> = "40"                     ;* image size
OUTPUT.AT<50,4> = "21"                     ;* image X pos
OUTPUT.AT<50,5> = "21"                     ;* image Y pos
```

Report Prefix

Prefix for report HTML & PDF files. By default the account name is used if the Report Prefix is blank. Slashes and dots will be removed. Leave blank to continue using the account name.

Secure Uploads Folder

If set to Yes then DesignBais will prevent unauthenticated access to the website *uploads* folder. Users will need to log in to the application before they upload or download files. If set to No then this feature is not active. The Inherit setting will use the global setting from Global Login Parameters.

Report Header Overflow

Overflow is hidden in report fields by default to prevent text from overlapping. You can set this field to to allow overflow in the header, column header and footer sections of reports. The setting may be inherited from Global Parameters.

Report px to mm Conversion

DesignBais Version 6 used a factor of 4.05 to convert px to mm. This was changed to 3.7795 for Release 7 and 8. You can set a different value by entry in this field. The default remains as 3.7795. This value can be left blank in which case the value in Global parameters, if any, will be used.

Earliest Date

The earliest date to be entered into the application.
May be a valid date or a calculation like T, T+nnn or T-nnn where T is the date at run time plus or minus a number of days.
This value will override the value in Global Parameters.
This is useful to prevent keying errors in the year that produce a 5 digit year number that is not handled correctly by a D4 date conversion or a date too far back in time. See Date Format.

Maximum Date

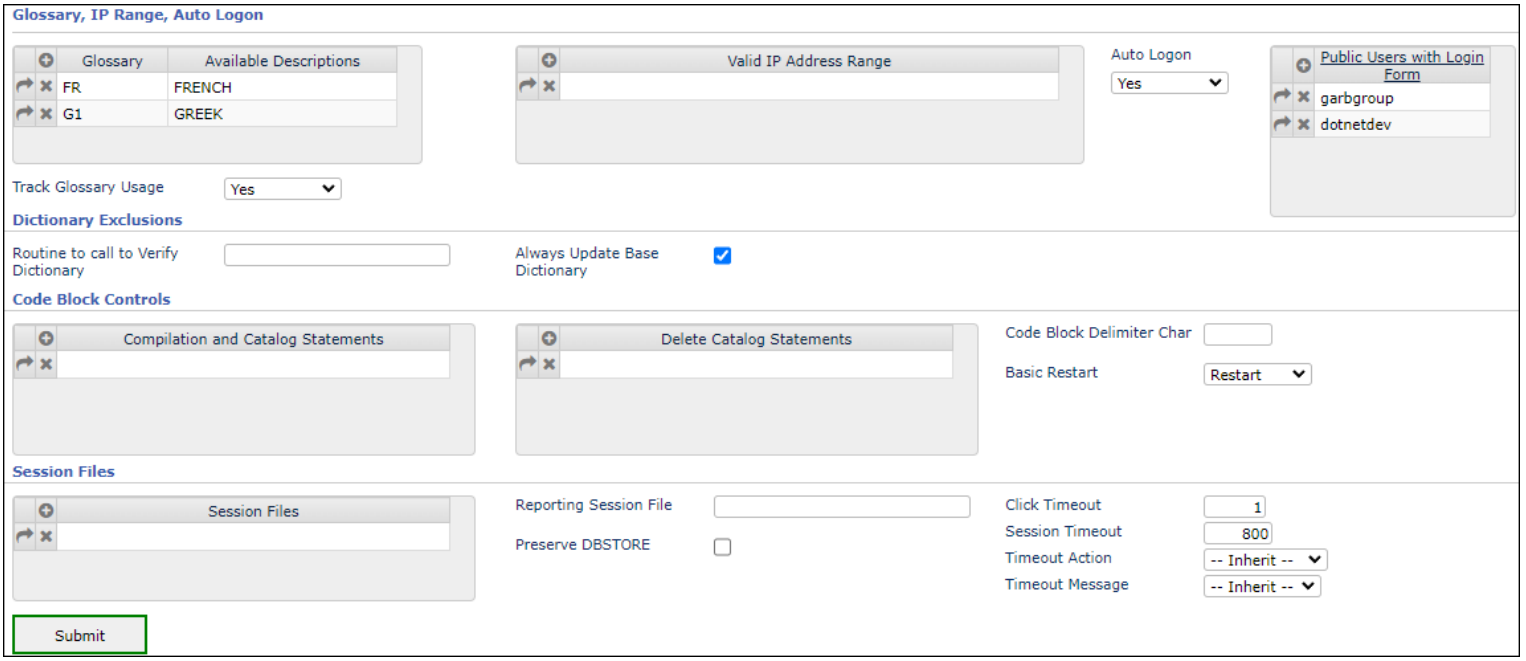
The maximum date to be entered into the application.
May be a valid date or a calculation like T, T+nnn or T-nnn where T is the date at run time plus or minus a number of days.
This value will override the value in Global Parameters.
This is useful to prevent keying errors in the year that produce a 5 digit year number that is not handled correctly by a D4 date conversion or a date too far back in time. See Date Format.

Before Screen Sequence

This setting will control when the BEFORE SCREEN subroutine attached to the menu structure is run. A setting here will override the Global setting.

Leave as *Inherit* if this account is to use the Global Parameters Setting.

Menu Only	the BEFORE SCREEN will only be called when a menu option is clicked. It will be before the form BEFORE DISPLAY is invoked.
First	the BEFORE SCREEN will be invoked whenever a new form opens (menu click or PROCESS.STACK). It will be called before the form BEFORE.DISPLAY.
V6 Mode	the BEFORE SCREEN will be invoked whenever a new form opens (menu click or PROCESS.STACK). It will be called after the form BEFORE.DISPLAY. This was the V6 behaviour.



Glossary Details

DesignBais has a terms-based glossary feature that will enable the set-up of multiple glossaries. Each user is set-up with a default glossary that will point them to the correct set of descriptions for the DesignBais account. The glossary name can be any text field, but it is recommended to keep the name short (without spaces), as it will be used for each definition written to the glossary.

Track Glossary Usage

The “Where Used” tracking feature of the glossary may be turned off using this field. The “Inherit” setting allows the Global Parameter setting to determine if “Where Used” tracking is operative.

Valid IP Address Range

The Valid IP Address Range determines what source IP addresses are valid for this System.
This range is used when no Valid IP Ranges is entered against a User.
If both User and System Valid IP Ranges are left blank than all IP addresses are allowed.
You can enter a partial IP address, a complete IP address or a range of IP addresses.
If a partial address is entered DesignBais will validate the part entered against the same part of the source IP address.

An example of an IP Range is 123.456.789.10-50 where the first 3 segments must match the start of the source IP address (123.456.789) and the last segment must be in the range (10 to 50).

IP6 addresses may be used. You must enter either all 8 ":" separated segments or the last 4. If all 8 are present then they will be matched to the source IP otherwise only the last 4 segments of the source IP will be used. Again a range may be indicated by using a "-" in the last segment.

Auto Logon

When starting a second or subsequent tab session on a browser the logon form can be by-passed and DesignBais will log on the same user as was logged on in the initial tab. This flag can be set to *Inherit* to inherit the equivalent setting from the Global Login Parameters. Refer to the *Global Login Parameters* section of this manual for more details.

Public Users with Login Form

The list of Public Login users that start in a login form where the operator switches to their own application user ID. This list is used by the Auto Login feature to automatically switch to the currently connected user when a new tab is opened in a browser.

Routine to Call to Verify Dictionary

This field provides the ability to name a subroutine to call to verify dictionary load and update processes. The named program will be invoked in the following manner.

CALL @YourProgramName(OPERATION,PROCESS.KEY,PROCESS.RECORD,PROCESS.FLAG)

OPERATION will be set to either 1, 2 or 3.

1 = Load Dictionary Operation
2 = Save Property Operation (Submit Button on the Field Properties Form)
3 = Delete Property Operation (Delete Button on the Field Properties Form)

PROCESS.KEY = Filename*Dictionary for OPERATION of 1
PROCESS.KEY = Filename*Dictionary for OPERATION of 2 (Provides Filename reference)
PROCESS.KEY = Filename*Dictionary for OPERATION of 3 (Provides Filename reference)
PROCESS.RECORD with OPERATION = 1. Contains the DBIPROP record details
PROCESS.RECORD with OPERATION = 2. Contains the Dictionary record details.
PROCESS.RECORD with OPERATION = 3. Contains the Dictionary record details.

PROCESS.FLAG will originally be set to 1 indicating that the dictionary will be processed.

OPERATION = 1

If the PROCESS.FLAG is returned in a non-true (zero or null) state the dictionary will not be loaded into DesignBais.

OPERATION = 2 or 3

If the PROCESS.FLAG is returned in a non-true (zero or null) state the dictionary will not be written or deleted. This provides the developer with the ability to modify dictionary record format before it is written to the database or not write the dictionary at all. See code sample below.

You can always add a DBDS entry to your program to display the items not loaded or dictionaries not written.
Eg. DBDS<-1> = PROPERTY.NAME:" not loaded"
This will be displayed at the end of the load. (Program must include DBI.COMMON to pass DBDS.)

Always Update Base Dictionary

If checked the database dictionary will always be updated. If not checked the database dictionary will not be updated if a version exists with a preceding period'. This feature is primarily intended for sites with SB+ extended dictionaries. To avoid updates to the database dictionary files without creating additional dict items, use a program in the Routine to Call to Verify Dictionary.' Eg:

```
01 SUB VERIFY.DICT(OPERATION, PROCESS.KEY, PROCESS.RECORD, PROCESS.FLAG)
02 INCLUDE DBI DBI.COMMON
03 PROCESS.FLAG = 0
04 RETURN
```

Compilation and Catalog Statements

This prompt allows for multiple statements to be executed whenever DesignBais is required to compile and catalog a program or a code block. If left bank, DesignBais will use default compile and catalog statements as performed pre-release 4. In your statements you can include %filename or %itemname to indicate where DesignBais is to place the Filename and Itemname for the compile.

Eg BASIC %filename %itemname
CATALOG %filename %itemname
RESTART.COMMUNICATIONS

(The last line of this example may be required for communication interfaces which need to be restarted in order to allow existing processes to use previously cached object code. There is no RESTART.COMMUNICATIONS command - another product-specific command may need to be used here.)

Delete Catalog Statements

This prompt allows for multiple statements to be executed whenever DesignBais is required to delete the catalog entry for a program or a code block. If left bank, DesignBais will use a default delete catalog statement.

In your statements you can include %filename or %itemname to indicate where DesignBais is to place the Filename and Itemname for the delete catalog.

E.g. DECATALOG %filename %itemname

Code Block Delimiter Character

The code block delimiter character is used in the key of code block source code records to separate the parts of the key, for example DBCLIENT%REPORT%DERIVED%DERIVED.1. By default, the code block delimiter character is a % sign. If you wish this to be different, please enter a single character, Eg. $.

Basic Restart

Optional connection restart after compiling in the DesignBais Code Editor:
Restart - restart current pooled connection
Restart All - restart all pooled connections
Recycle - recycle the IIS Application Pool - not recommended as connections are not dropped gracefully.
Note: It is not recommended that Code Editor compilation is used in production environments while users are processing as this can lead to a session abort.

Sessions Files

From release 4.2 onward, you may introduce load balancing to the session file for large sites. This provides the ability to localise session files at an account (or account grouping) level. It also provides the ability to create multiple session files to further reduce I/O load on this file. Each user is assigned a unique session id. Each session id is then hashed to determine the session file to be used.

Reporting Session File

The creation of a sessions file for the temporary storage of reports is recommended. This will remove pressure from the sessions files in periods of peak activity since reports typically produce large amounts of output.

Preserve DBSTORE

By default DBSTORE is cleared at login and after a change of database account. If set to 'Yes' the *Preserve DBSTORE* option will attempt to restore DBSTORE values after an initial logon or change of account. If multiple sessions files are in use then, for this option to work, they must be shared across accounts and listed in the same order.

This parameter needs to be set in the target account. If "Preserve DBSTORE" is set then DBSTORE is read and written to the new session ID from both Hit 1 IDPrevWindow & Hit 3 IDWindowOld. It is then available in the target account.

Note that if the application is structured to logto an account, process, then return to the original account then both accounts become, in turn, the "target account" and the *Preserve DBSTORE* flag should be set in both.

Click Timeout

When DesignBais is re-setting the page in the browser with javascript it became necessary to introduce a delay in milliseconds to be sure that any click() event will take effect. If no value is available then a 1 msec delay will be used. A sample javascript command:
setTimeout(function(){document.getElementById("button5v1").click();}, 1);

Session Timeout

A session will expire after an idle period of the session timeout in minutes. Valid values are 0 to 1200 minutes (20 hours). The recommended value is 20 minutes or longer. A blank value or 0 will indicate that the Global Login Parameter Session Timeout is to be utilised. A default value of 20 minutes will be used if no Session Timeout has been supplied. See **DesignBais Page Refresh**, in the **DBIGLOBAL Login Parameters** section of this manual, which provides more detail.

Timeout Action

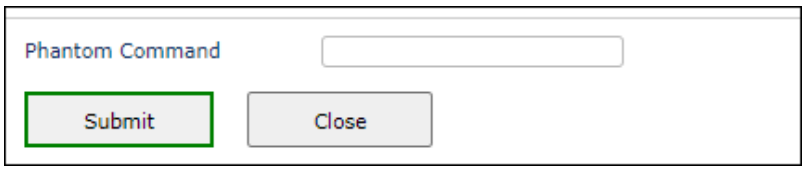
This setting determines the action to take when a DesignBais browser tab session has been idle for the Session Timeout period. *Inherit* will use the Global Login Parameter setting with the default being start a new session. *New Session* will display a timeout message if the user attempts to activate the session. The timeout warning message may be suppressed. The user will be automatically taken to their start form. *Logout* will take the user to the logout page. *Error Page* will take the user to the DesignBais error page.

Timeout Message

When a DesignBais browser tab session has been idle for the Session Timeout period then a timeout message will display if the user attempts to activate the session. The timeout warning message may be suppressed by selecting No. The user will be automatically taken to their start form. May be overwritten for individual users. A value entered here will overwrite the Global Parameters setting.

Phantom Command

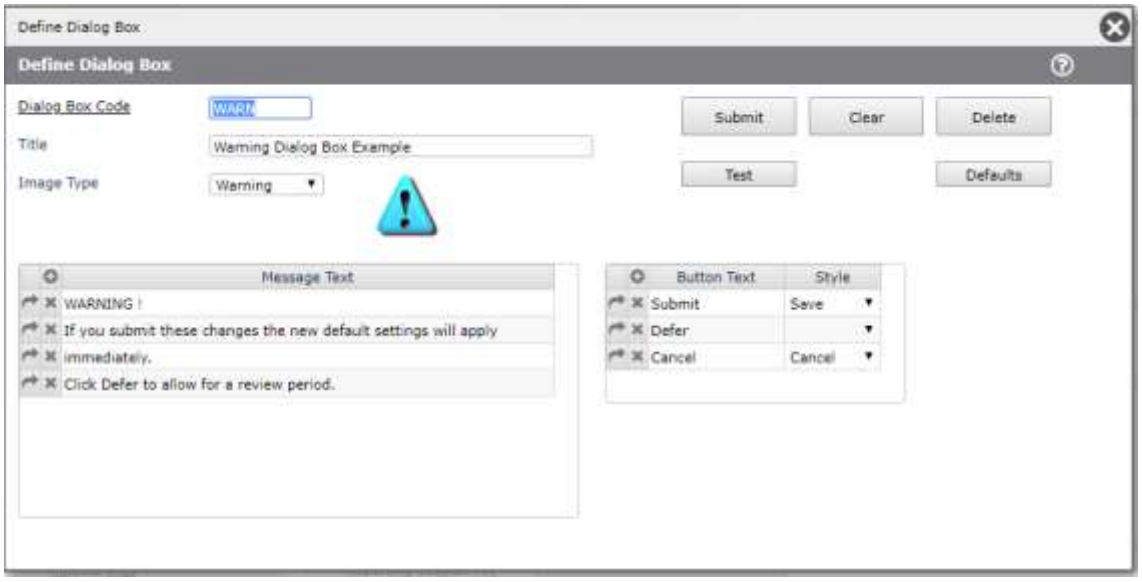
On D3 only the phantom command can be specified. The default value is "z". If the alternative "zs" or "zsd" is required then enter the value in this field.


Phantom Command
Submit
Close

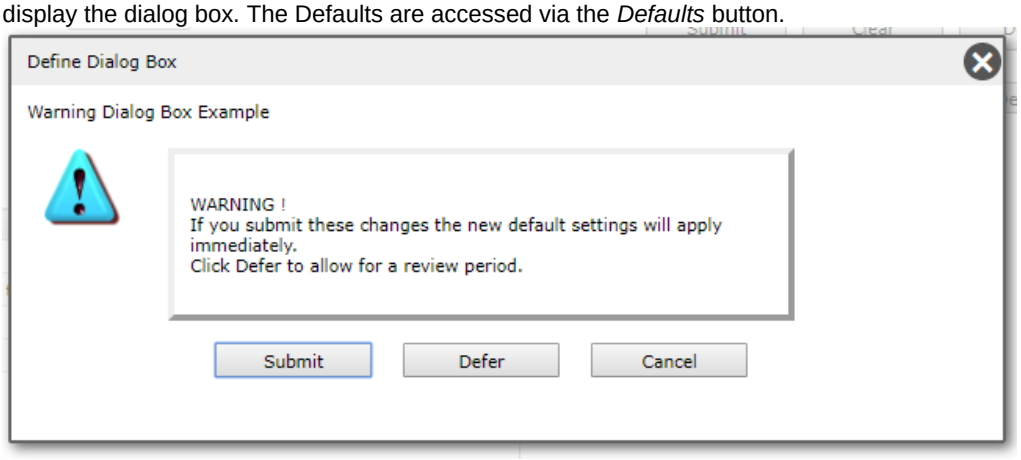
Dialog Setup

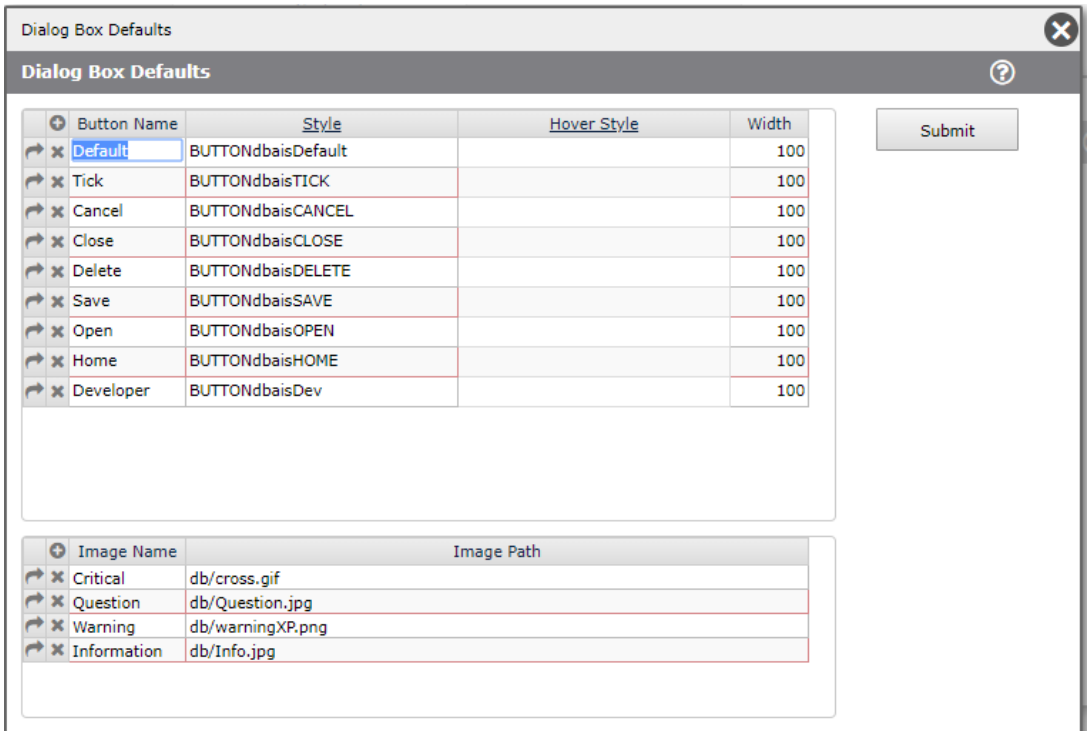
Dialog Setup

This option opens the new Dialog Box maintenance form.



This form allows the developer to create and save standard dialog box responses. Use the *Test* button to display the dialog box. The Defaults are accessed via the *Defaults* button.





It is also possible to create custom dialog boxes within basic code. The DBDEMO_DBOX form in the DesignBais Demonstration Forms includes example code.

You will need to include the equates for DBIPARMS and DBIGLOBAL in your basic routine.

```
$INCLUDE DBINET E.DBIPARMS
$INCLUDE DBINET E.DBIGLOBAL
```

The DBIGLOBAL equates include the following:

```
[EQU DBIGO.DBOTHER.DBOX.CONTROL                          TO    91  ;* Dialog Box Entry Points]
```

In this example the dialog box is initiated ffrom a button click event. In a real example you may be displaying the dialog in a validation event.

```
*
BUTTON:
*
      BEGIN CASE

         CASE SCREEN.NO = "DBOX" AND EVENTSOURCE = "B.DBOX"
            GOSUB DBOX.DISPLAY
```

The DBOX.DISPLAY paragraph supplies the details to be passed to the DesignBais DBI.G.DBOX subroutine.

```
*
DBOX.DISPLAY:
*
      DBOX.REC = ''
      DBOX.REC<DBIPM.DLOG.TITLE> = 'New Dialog Box Test'
      MSG = 'Client: <b>':DBRECORD<DBC.CLIENT.CODE>:' ':DBRECORD<DBC.CLIENT.NAME>:'</b>'
      MCNT = 3
      MSG<1,MCNT> = 'Click your button of choice'
      DBOX.REC<DBIPM.DLOG.BUTTON.TEXT> = 'Save':VM:'Save & Continue':VM:'Return':VM:'Quit'
      DBOX.REC<DBIPM.DLOG.BUTTON.STYLE> = '':VM:'':VM:'':VM:''
      DBOX.REC<DBIPM.DLOG.MSG> = MSG
```

```
      DBOX.REC<DBIPM.DLOG.IMAGE> = 'Question'
      DBOX.REC<DBIPM.DLOG.PARAMETER> = 'DBOX.PARAM'
      DBOTHER.RECORD(DBIGO.DBOTHER.DBOX.CONTROL) = DBOX.REC
      CALL DBI.G.DBOX(1,'','DBOX.PARAM')
      RETURN
```

The DesignBais DBI.G.DBOX subroutine reads the DBIPARMS_DBOX form, modifies it as directed by the contents of DBOX.REC passed via DBOTHER.RECORD(91), then writes it to DBISESSIONS and sets DBFORMLOCAL.

```
   SUBROUTINE DBI.G.DBOX(ENTRY.POINT,MSGPARAM,DLOGPARAM)
* Routine to set up Dialog Box display.
*
* DBIPARMS DBOX form is manipulated to display and fit required buttons.
* The base form is modified and written to DBISESSIONS before the form
* is invoked.
*
* Display correct image and formats the message text for HTML.
*
* ENTRY.POINT input  =   1 - DBOTHER.RECORD(91) available
*                               2 - DBIPARMS key in MSGPARAM
*                               3 - Message text in MSGPARAM
*                               4 - Message text in MSGPARAM to display on a
*                                           normal base form with no buttons.
* MSGPARAM    input     = blank for Entry point 1
*                               = DBIPARMS key for entry point 2
*                               = Message text for entry point 3 or 4
* DLOGPARAM   input     = Parameter to indicate where dialog was invoked
* DBOTHER.RECORD(91) input  = Dialog Box record details
*                               output = DBIPM.DLOG.SELECTED contains button clicked
*
* ************************************************************************
```

The MODAL RETURN event following the return from the display of the dialog box uses the entry point defined by the 3rd argument passed to DBI.G.DBOX (DLOGPARAM in the above example).

```
*
MODAL.RETURN:
*
         CASE SCREEN.NO = 'DBOX' AND EVENTSOURCE = 'DBIPARMS_DBOX'
            * Dialog Box
            * Parameters in DBOTHER.RECORD(DBIGO.DBOTHER.DBOX.CONTROL)
            DBOX.REC = DBOTHER.RECORD(DBIGO.DBOTHER.DBOX.CONTROL)
            ENTRY.POINT = DBOX.REC<DBIPM.DLOG.PARAMETER>
            BTN.CLICK = DBOX.REC<DBIPM.DLOG.SELECTED>
            BTN.TXT = DBOX.REC<DBIPM.DLOG.BUTTON.TEXT,BTN.CLICK>
            BEGIN CASE
               CASE ENTRY.POINT = 'DBOX.PARAM'
                  GOSUB DBOX.PARAM
               CASE 1
                  * No entry point = plain message, just close
            END CASE
            DBOTHER.RECORD(DBIGO.DBOTHER.DBOX.CONTROL) = ''   ;* clear dialog box
      END CASE
      RETURN
```

Control is passed to a paragraph named DBOX.PARAM (this can be any name as defined by the developer).

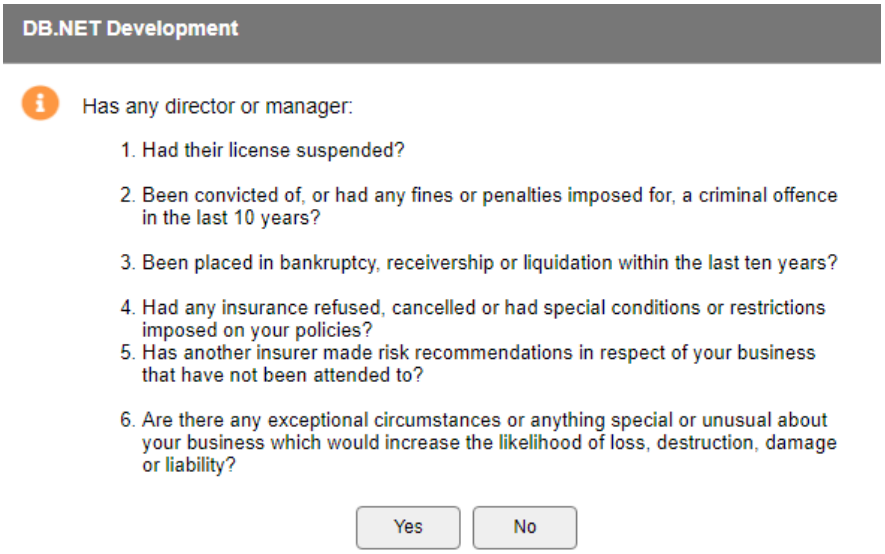
This paragraph determines which button was clicked and takes corresponding action as determined by the developer.

```
*
DBOX.PARAM:
*
      START.STR = 'Well done. You clicked the '
      END.STR = ' button!'
      IF DBMODAL.CLOSED.VIA.X THEN
         BTN.TXT = '"X"'
         START.STR = 'You closed the form via the '
      END ELSE
         BEGIN CASE
```

```
      CASE BTN.TXT = 'Save'
         NULL
      CASE BTN.TXT = 'Save & Continue'
         NULL
      CASE BTN.TXT = 'Return'
         NULL
      CASE BTN.TXT = 'Quit'
         NULL
   END CASE
END
DISP.TEXT  = START.STR:"<span Style='background-color:thistle'>"
DISP.TEXT := BTN.TXT:"</span>":END.STR
DBWORK<DBC.WORK1.WK> = DISP.TEXT
RETURN
```

You can use the <li tag to format dialog messages. Formating can also be enhanced with a display class:

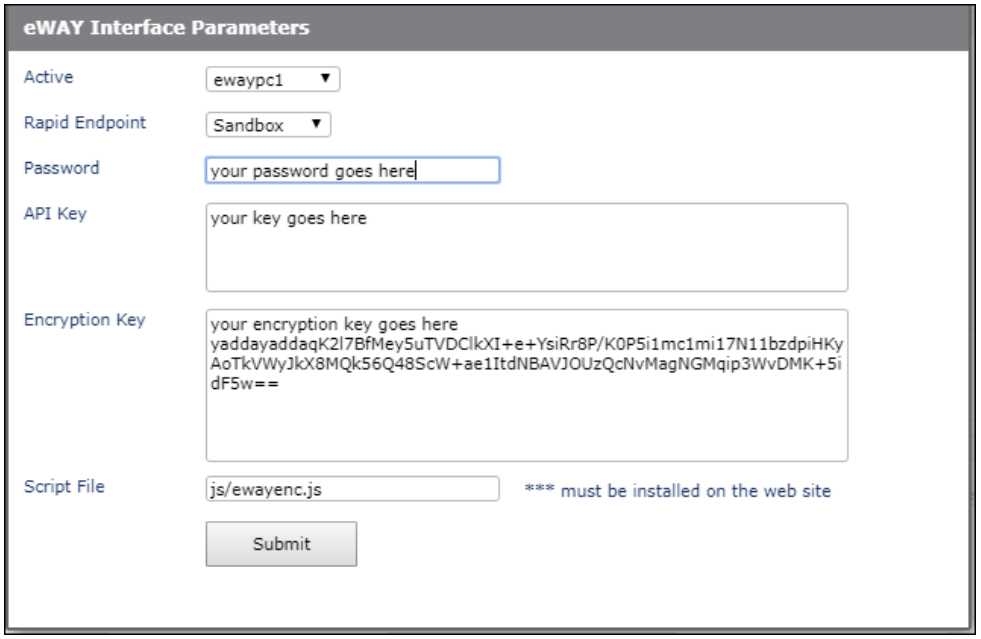
```
Has any director or manager:<br><ol><li class="JLTEST">Had their license suspended?<br><br></li><li class="JLTEST">Been convicted of, or had any fines or penalties imposed for, a criminal offence in the last 10 years?<br><br></li><li class="JLTEST">Been placed in bankruptcy, receivership or liquidation within the last ten years?<br><br></li><li class="JLTEST">Had any insurance refused, cancelled or had special conditions or restrictions imposed on your policies?</li><li class="JLTEST">Has another insurer made risk recommendations in respect of your business that have not been attended to?<br><br></li><li class="JLTEST">Are there any exceptional circumstances or anything special or unusual about your business which would increase the likelihood of loss, destruction, damage or liability?</li></ol>
```



eWAY

eWAY

Click to open the eWay Interface Parameters form. This provides a means of delivering payment gateway functionality which is PCI compliant. The goal of PCI compliance is to ensure that merchants provide the maximum security when processing customer payments or handling customer data.



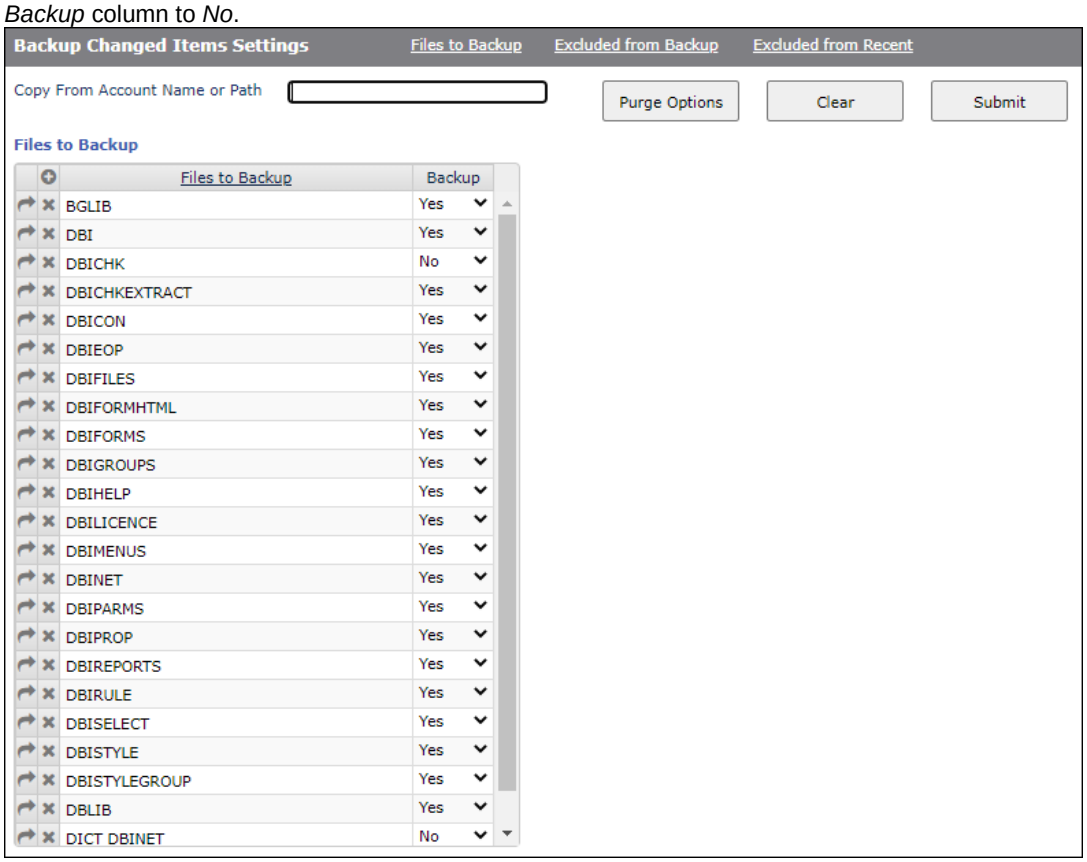
Backup Settings

Backup Settings

This option opens the Backup Changed Items settings form. Backup Changed Items is a utility to backup selected items to DBIBACKUP. Backup of an item occurs if the current version of the item differs from the most recent backup version of the item.

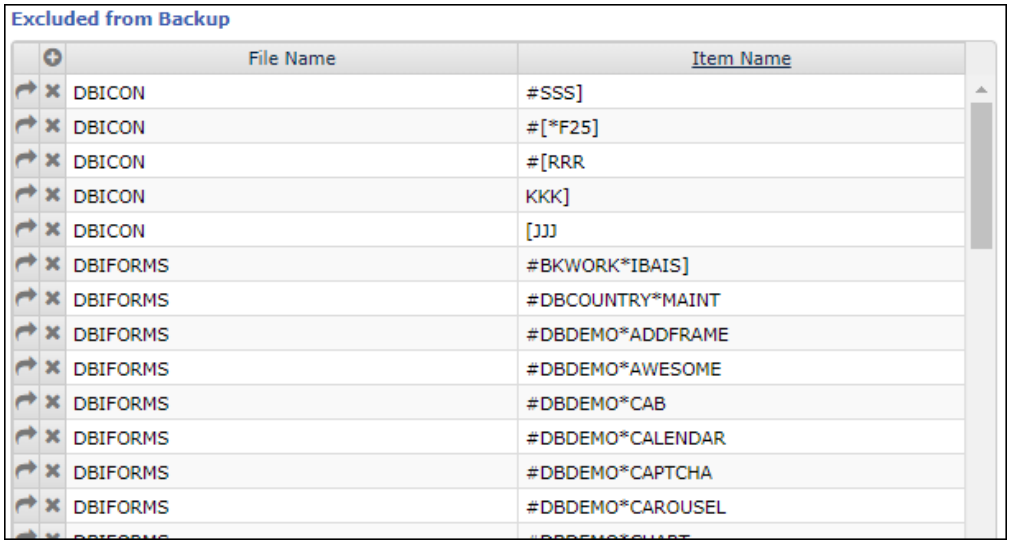
Files to Backup

The list of files that are to be backed up. The backup of a file in this list can be disabled by setting the *Backup* column to *No*.



Excluded from Backup

A list of files and items that are excluded from the backup. The file name entered here may be the name of a file from the list of Files to Backup or it may be the name of a file that is part of the key of an item that is held on one of the files in the list of Files to Backup.



If the Item Name is left null or contains an "*" then all items for this file will be excluded. Otherwise only items matching the specified item name will be excluded. Wild card characters "[" and "]" can be used at the start and end of the item name if required.

A file name must be entered in the associated excluded file name field before the item name can be entered.

Prefix the item name with the hash character (#) to designate that all items are to be excluded apart from those matching this item name. For example "#REPORT]" causes all item ids other than those commencing with "REPORT" to be excluded. You may have multiple “#string]” entries for the same file. All entries are evaluated together at run time so that all items designated by the “#” entries are backed up and all other items are excluded.

Excluded from Recent

A list of files and items that are excluded from the display of recent backup items. The file name entered here may not be the name of the file that is being backed up - it may be the name of a file that is part of the key of an item that is held on one of the files in the list of files to Backup.

If the associated Item Name is left null or contains an "*" then all items for this file will be excluded. Otherwise only items matching the specified item name will be excluded. Wild card characters "[" and "]" can be used at the start and end of the item name if required.

A file name must be entered in the associated excluded file name field before the item name can be entered.

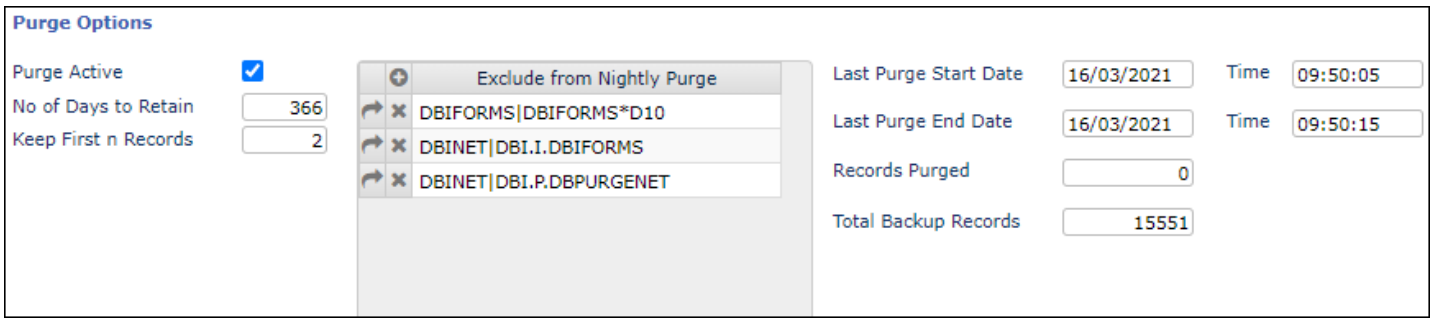
Prefix the item name with the hash character (#) to designate that all items are to be excluded apart from those matching this item name. For example "#REPORT]" causes all item ids other than those commencing with "REPORT" to be excluded.

Copy Excluded from Backup list

The list of file names and item names from the *Excluded from Backup* list will be appended to the current contents of the *Excluded from Recent* list.

Purge Options

Specify settings to control the purging of items from the backup file. Purging is performed by the DesignBais nightly purge routine.



Purge Active Toggle this flag toenable and disable purging.

No of Days to Retain

Specifies the number of days that backup items remain on the backup. After this number of days items will be purged.

Keep First n Records

This option allows you to retain the first ever backup of an item independently from the setting in *No of Days to Retain*.

Exclude from Nightly Purge

This list specifies items that are not to be purged. Entries in this list must be in one of the following forms where the pipe character is used to separate the file and record names:

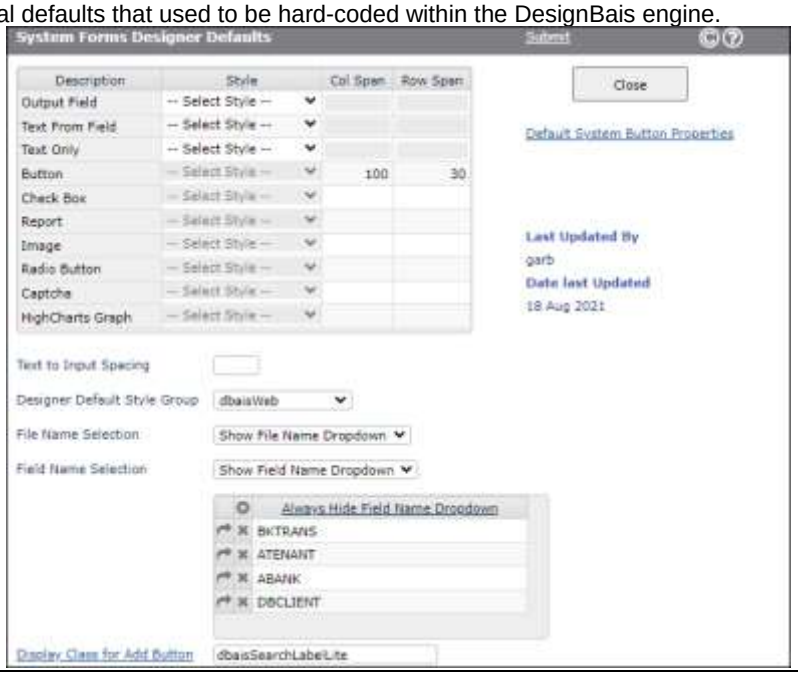
Filename
DICT Filename
Filename|Recordname
DICT Filename|Recordname

Designer Defaults

Designer Defaults Use this form to enter default values for use in Forms Designer.

From Release 8.3.3.6 the Forms Designer tool allows form elements to be sized based on the the styles defined in the style group designated for the form.

These default style, column span, row span and input spacing values are used when the form element style does not define a default value. If these default values are blank then Forms Designer uses the traditional defaults that used to be hard-coded within the DesignBais engine.



Designer Default Style Group

Set the name of the Style Group that is to be used as the default for new forms in the Forms Designer tool. If set to *Inherit* then the value set in General Global Parameters will be used.

Default System Button Properties

This option allows the developer to define a set of buttons that are commonly required on forms. These defaults are available to all accounts that share the System Parameters held on DBIPARMS. A similar set of defaults can be defined for each Style Group. Developers using Forms Designer can access both these sets of default buttons. A default button can be selected, the properties amended as required, and then added to a form

From Release 8.5.1.6 the following default options are available:

File Name Selection

In Forms Designer the *File Name* field can be selected from a dropdown selection list, as well as from a *File Name* selection process. On systems with a large number of files the creation of the dropdown list can impact the time it takes to render the DesignBais tools form.
To overcome this delay developers can now choose to hide the File Name dropdown selection list.

Field Name Selection

In Forms Designer the *Field Name* field can be selected from a dropdown selection list, as well as from a *Field Name* selection process. If a file has a large number of field properties the creation of the dropdown list can impact the time it takes to render the DesignBais tools form.
To overcome this delay developers can now choose to hide the Field Name dropdown selection list.

Always Hide Field Name Dropdown

This option is provided for developers who wish to retain the Field Name dropdown selection list for most files but need to hide this dropdown for particular files. Enter the name of files that have a particularly large number of field properties. Forms designer will hide the field name dropdown list for forms based on any of the files listed in this field.

.

Amazon SNS

Amazon SNS

This option opens a form to maintain Amazon Simple Notification Service (SNS) parameters. The parameters are held on the DBIPARMS file with a Record Id of 'SNS'.

If the DBIGLOBAL file is present in DesignBais then the SNS parameters may be entered at the global level and used for all accounts in which there is no active SNS record in the local DBIPARMS file. Set the active flag off in a local account in order to force DesignBais to reference the DBIGLOBAL settings.

Refer to the DBIGLOBAL File section in this Reference Manual, under the Amazon SNS heading for documentation of the SNS parameters.

DBMail

DBMail

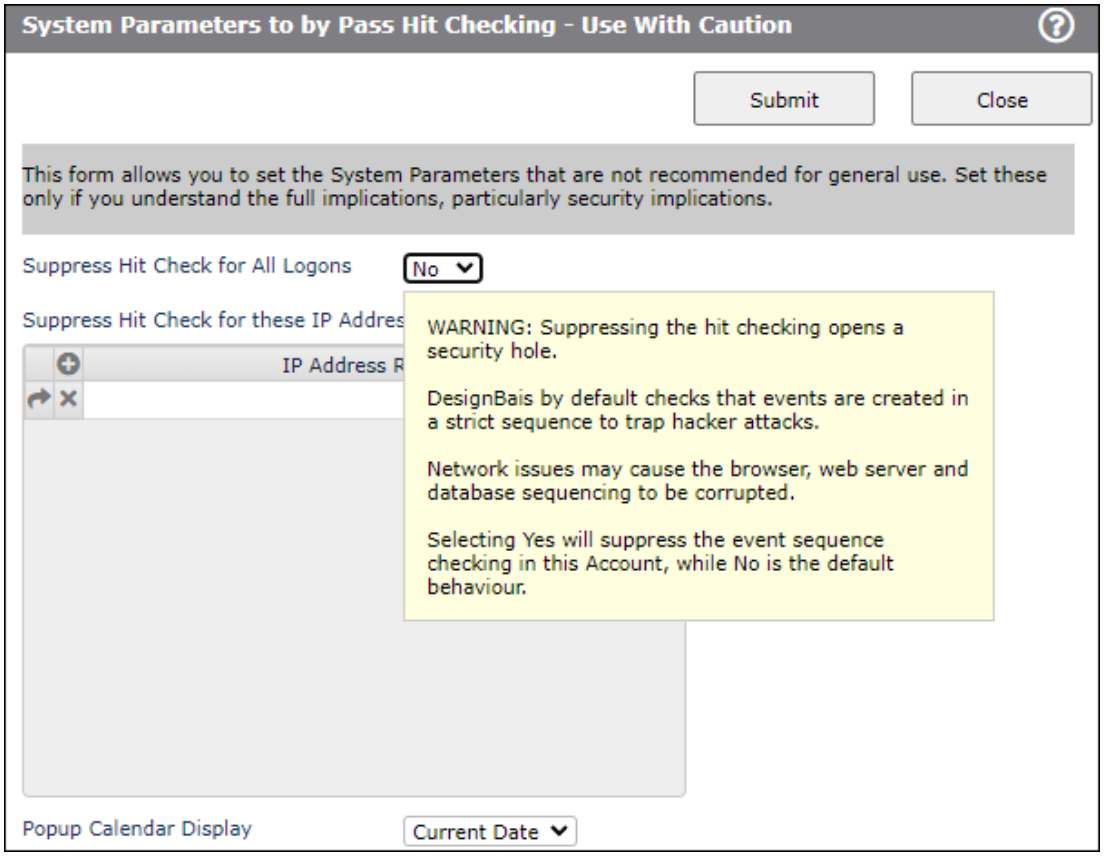
Click to open the DBMail Configuration form. Refer to the *DBMail* section of this Reference Manual.

Admin

You must be a member of the DBAdministrator Group to see these options

Suppress Hit Check for All Logons

This setting must be used with extreme caution. Suppressing the hit checking carried out by DesignBais opens a security hole that can be exploited by malware. This setting should only be set to *Yes* if there are environmental / network issues causing DesignBais hits to arrive out of sequence.



Suppress Hit Check for these IP Addresses Only

The No Hit Check IP Address Range determines what source IP addresses will by pass the standard hit checking.

You can enter a partial IP address, a complete IP address or an IP address range.

If a partial address is entered DesignBais will validate the part entered against the same part of the source IP address.

A sample of an IP Range is 123.456.789.10-50 where the first 3 segments must match the start of the source IP address (123.456.789) and the last segment must be in the range (10 to 50).

IP6 addresses may be used. You must enter either all 8 ":" separated segments or the last 4. If all 8 are present then they will be matched to the source IP otherwise only the last 4 segments of the source IP will be used. Again a range may be indicated by using a "-" in the last segment.

Popup Calendar Display

The Popup Calendar can be set to display either the current date or the date value in the database field.

F – Field Value

D – Current Date

This setting is stored in the DBIPARMS record id 0 (zero).

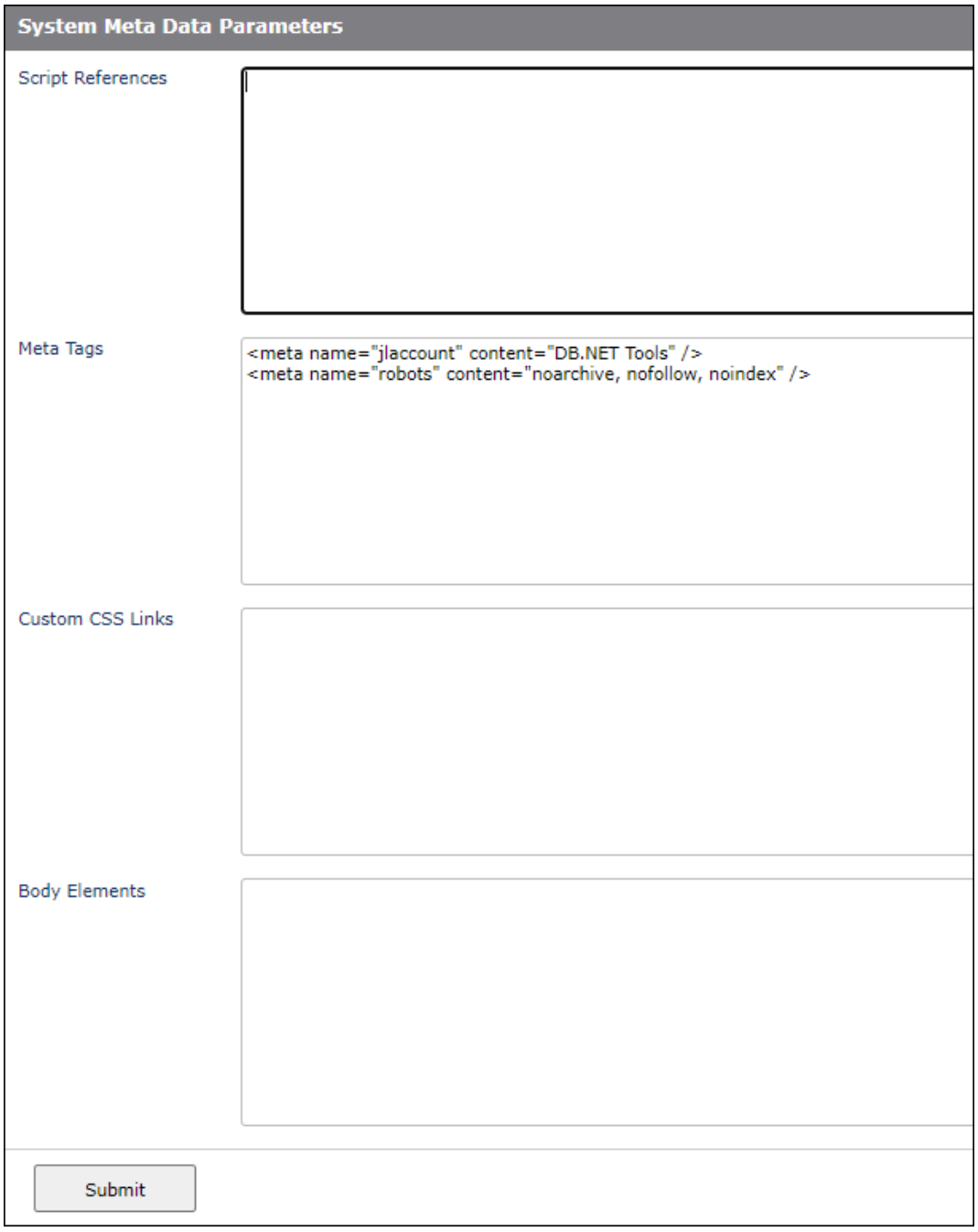
Meta Data

System Meta Data Parameters

Click to open the System Meta Data Parameters maintenance form.

This has been named in order to relate to the “Global Meta Data Parameters” where similar parameters may be entered for all linked accounts. This form allows the Global Parameters be overridden for a particular account.

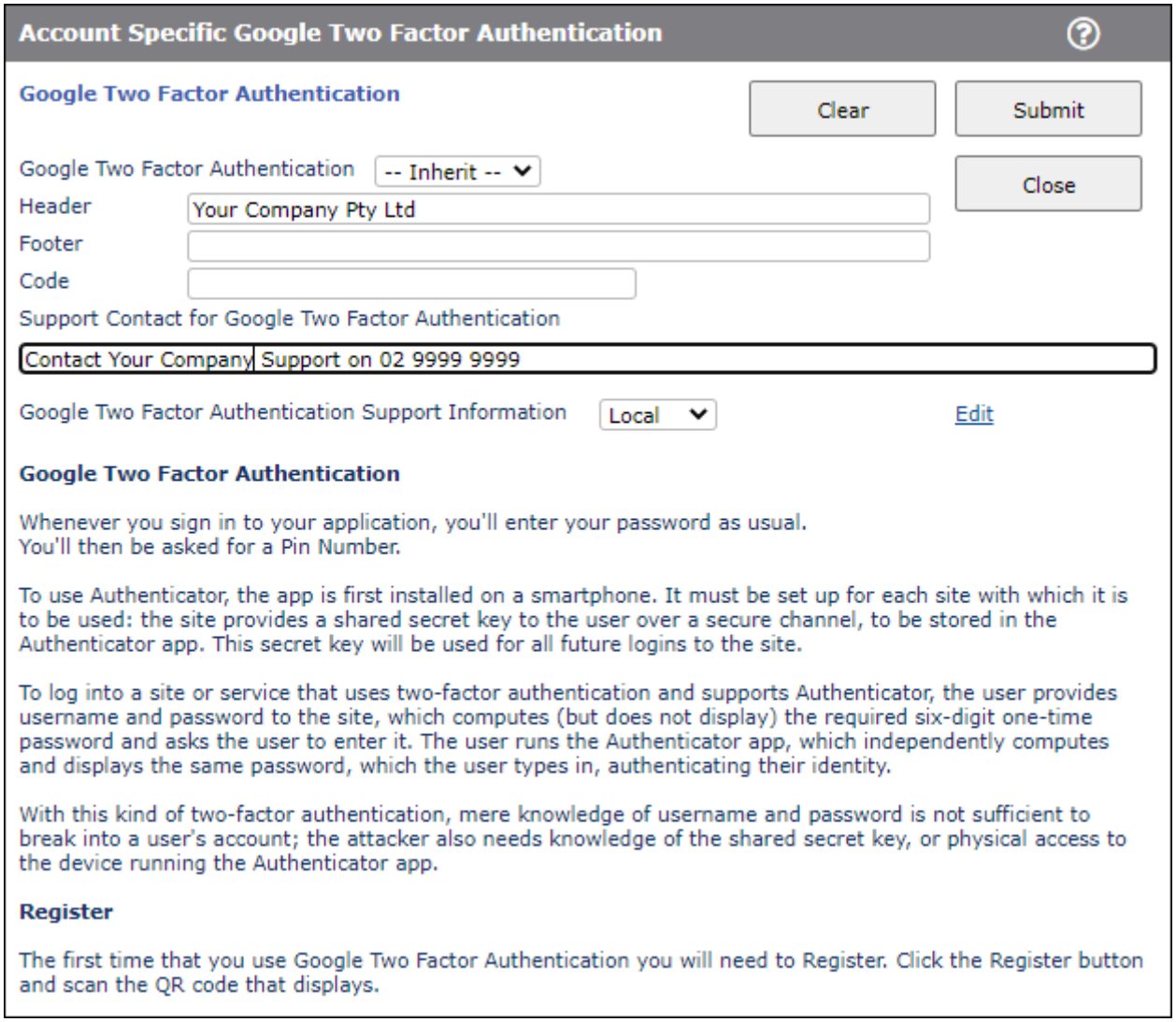
See the Global Parameters maintenance for an explanation of the Script References, Meta Tags, Custom CSS Links and Body Elements entry fields.



Google Authorisation

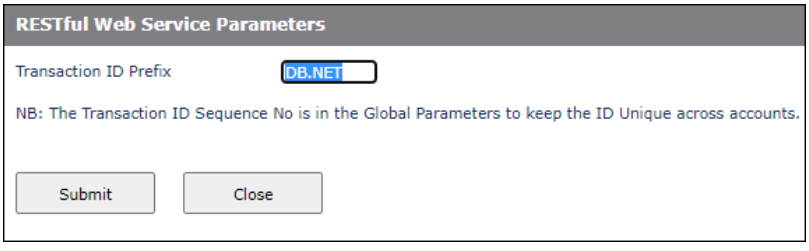
Google Two Factor Authentication

Refer to the Global Parameters settings for more details.



RESTful Web Service

Refer to DBCALLWS in this manual.



The RESTful web service interface will log errors with a 16 character Transaction ID consisting of this 8 character prefix followed by an 8 digit sequential number.

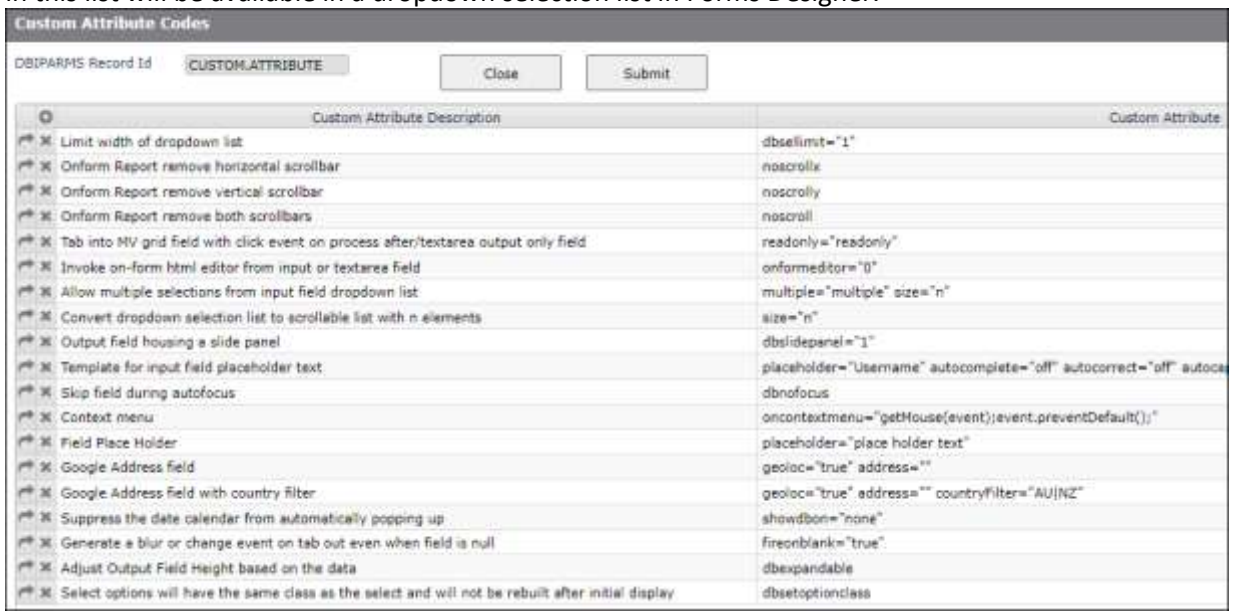
The default prefix will be the first 8 characters of the account name.

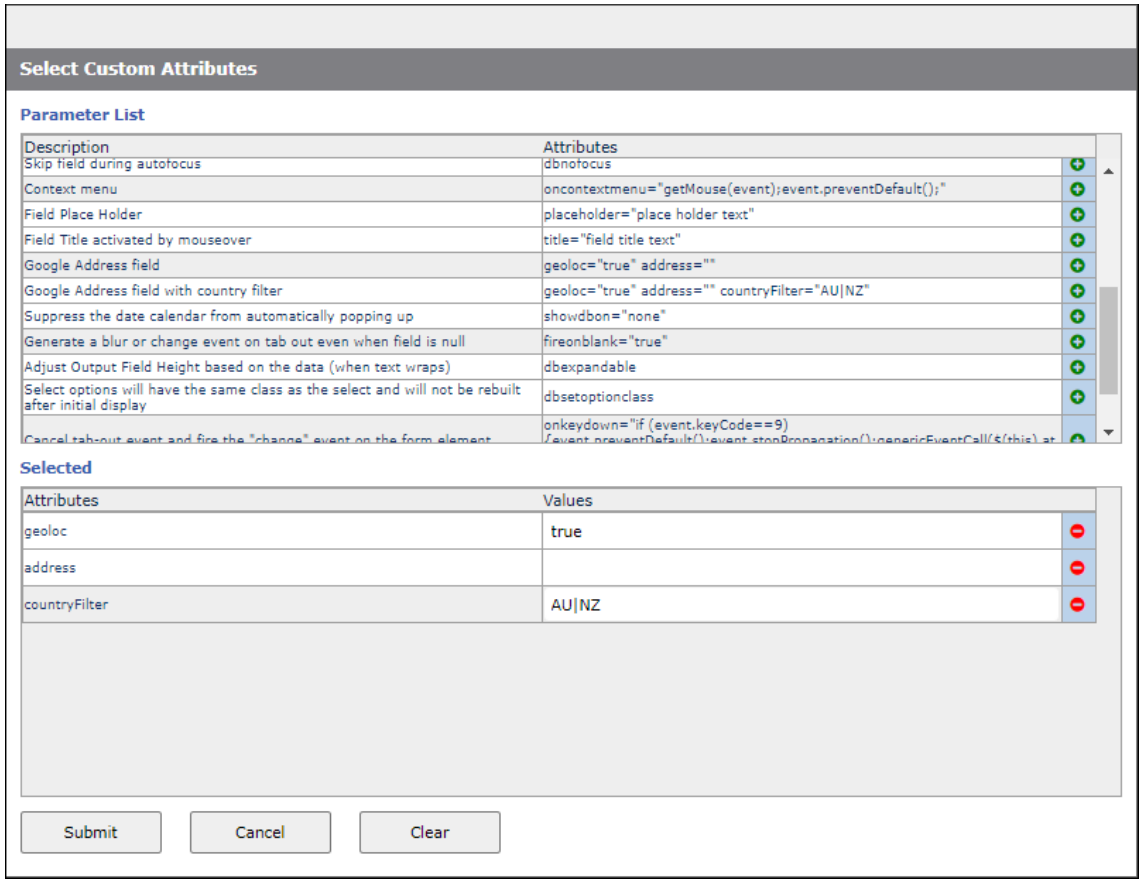
The prefix will be padded to 8 characters if necessary by using a suffix of zeros.

Custom Attributes

The main use of *custom attributes* is to add your own attributes to a field in order to allow some sort action via javascript.

The DesignBais custom attributes are displayed in this maintenance form. Developers can add to this list. All attributes in this list will be available in a dropdown selection list in Forms Designer.





The list of custom attributes available in the dropdown are shown here:

Limit width of dropdown list	dbsellimit="1"
Onform Report remove horizontal scrollbar	noscrollx
Onform Report remove vertical scrollbar	noscrolly
Onform Report remove both scrollbars	noscroll
Tab into MV grid field with click event on process after/textarea output only field	readonly="readonly"
Invoke on-form html editor from input or textarea field	onformeditor="0"
Allow multiple selections from input field dropdown list	multiple="multiple" size="n"
Convert dropdown selection list to scrollable list with n elements	size="n"
Output field housing a slide panel	dbslidepanel="1"
Template for input field placeholder text	placeholder="Username" autocomplete="off" autocorrect="off" autocapitalize="off" spellcheck="false"
Skip field during autofocus	dbnofocus
Context menu	oncontextmenu="getMouse(event);event.preventDefault();"
Field Place Holder	placeholder="place holder text"
Field Title activated by mouseover	title="field title text"
Google Address field	geoloc="true" address=""
Google Address field with country filter	geoloc="true" address="" countryFilter="AU|NZ"
Suppress the date calendar from automatically popping up	showdbon="none"
Generate a blur or change event on tab out even when field is null	fireonblank="true"
Adjust Output Field Height based on the data	dbexpandable
Select options will have the same class as the select and will not be rebuilt after initial display	Dbsetoptionclass
Cancel tab-out event and fire the "change" event on the form element	onkeydown="if (event.keyCode==9){event.preventDefault();event.stopPropagation();genericEventCall($(this).attr(\'id\'),\'change\')}"

Custom attributes are appended to the xml. So if DesignBais does not already send a particular attribute it is *likely* that you can send it as a custom attribute.

In the DBIGLOBAL_D21 shown below DesignBais does not set the *placeholder* attribute so it is set via *custom attributes* and the DesignBais database component adds it to the xml that is sent to the browser.

You cannot use *custom attributes* to set the class of a field. It is ignored, meaning that the DesignBais class, assigned via Forms Designer, remains in place.

From Release 7.5 onwards the DesignBais subroutine DBI.G.AJXCMD can be used to set style attributes.

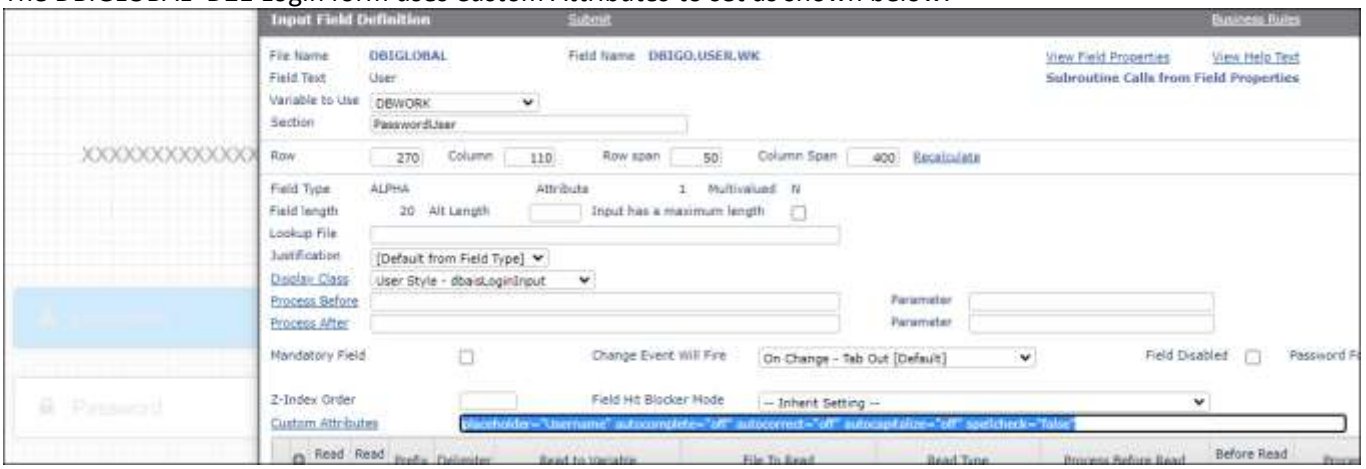
Example of Usage:

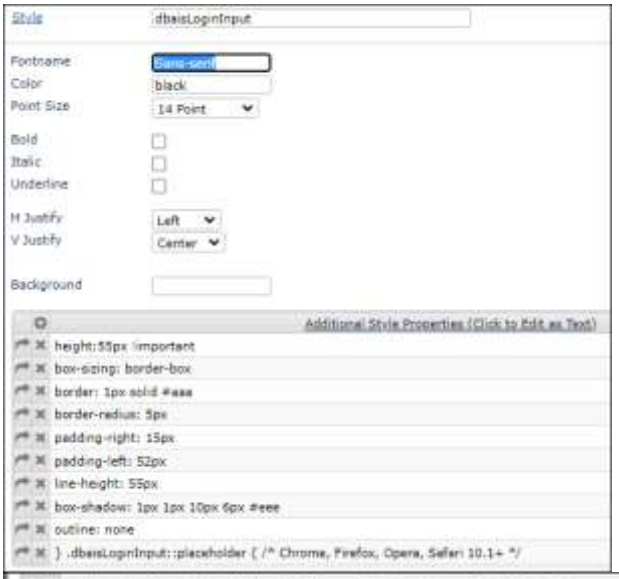
```
IF DBVALUE = 'N' THEN
   CLR1 = 'thistle'; CLR2 = 'yellow'; WID3 = 'thick'
END ELSE
   CLR1 = '#8e578e'; CLR2 = '#ff4d4d'; WID3 = 'thin'
END
AJXCMD = 'SS';* rdSetStyle
AJX.ARG = ''
AJX.ARG<1,-1> = 'd948937'
AJX.ARG<2,-1> = 'element'
AJX.ARG<3,-1> = 'background-color':SVM:'border-width'
AJX.ARG<4,-1> = CLR1:SVM:WID3
AJX.ARG<5,-1> = 0               ;* Millseconds Delay 0-1000 is not used by rdSetStyle
*
AJX.ARG<1,-1> = 'd978206'
AJX.ARG<2,-1> = 'element'
AJX.ARG<3,-1> = 'color':SVM:'background-color':SVM:'border-width'
AJX.ARG<4,-1> = 'red':SVM:CLR2:SVM:WID3
AJX.ARG<5,-1> = 0               ;* Millseconds Delay 0-1000 is not used by rdSetStyle
```

```
CALL DBI.G.AJXCMD(AJXCMD,AJX.ARG)
```

See List of custom attributes used by DesignBais Tools.

The DBIGLOBAL*D21 Login form uses *Custom Attributes* to set as shown below:



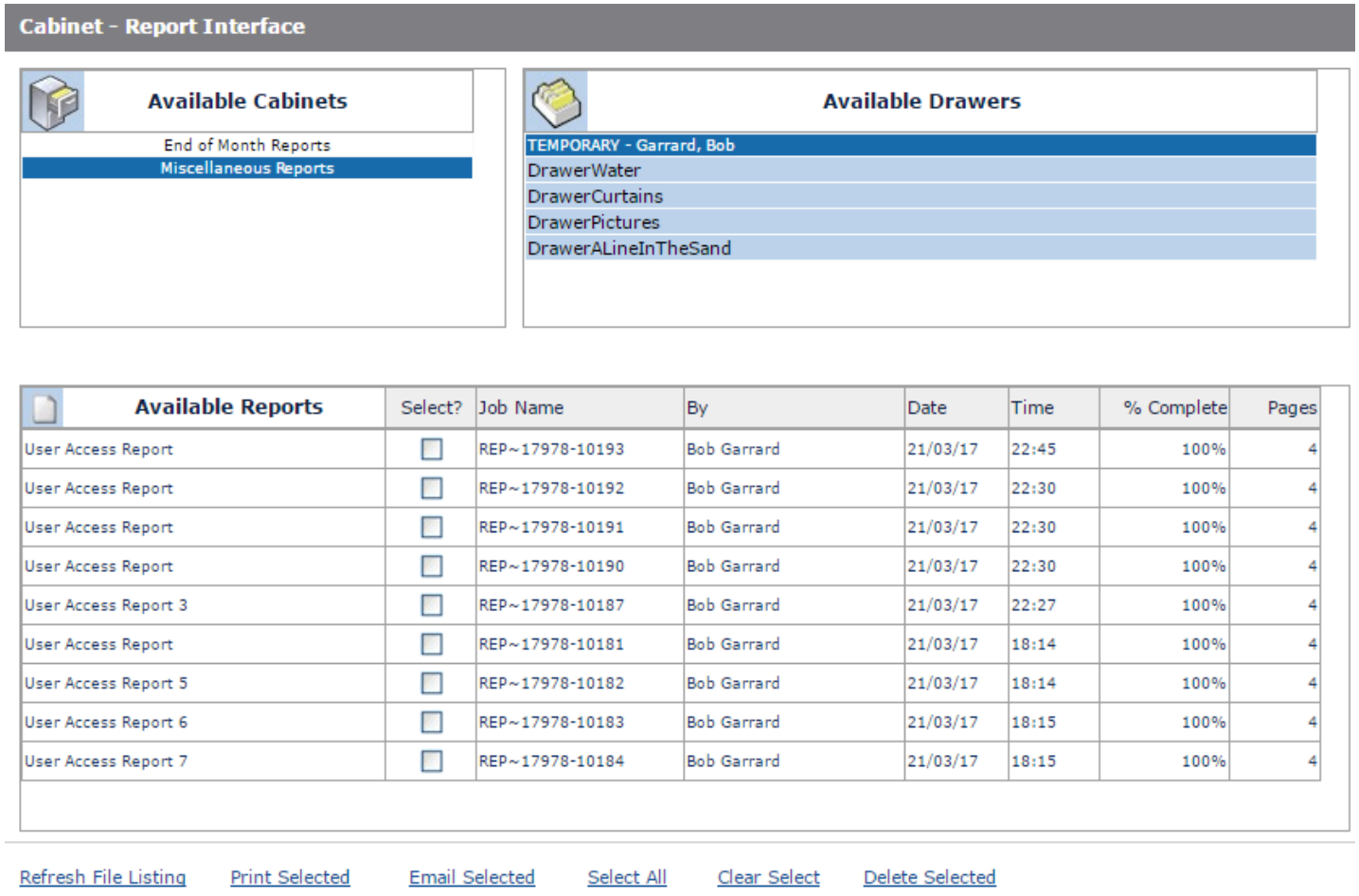


Cabinets

Cabinets provide the end-user the ability to store reports into a filing-cabinet and a draw. Each report can be named to make it easier to find in the future. Cabinet maintenance is accessed from the System Parameters menu.

Reports can also be stored in Cabinets programmatically. This provides a mechanism for end-of-period reporting processes to occur.

Example View of Cabinets and the contents of the Temporary Drawer:



There are three main sections to the Cabinet view.

The Cabinet View

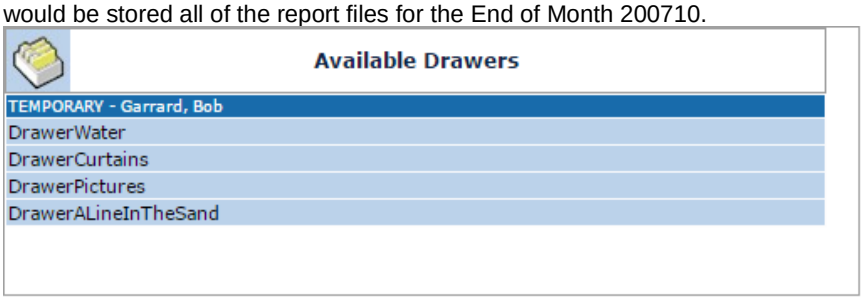
Cabinets are assigned in the eXpress set-up form.

Each cabinet represents a physical file on the database. Click on the required cabinet to display the list of drawers available.

Cabinets may be set-up with restricted access. This provides security for reporting.

The Drawer View

A cabinet may contain many drawers. In this example (left), a drawer has been created to store End of Month reports. In this drawer would be stored all of the report files for the End of Month 200710.



Drawers may be set-up with restricted access. This provides security for reporting. A temporary drawer will be set-up for each user as they save reports, or run reports to cabinets. Temporary drawers may only be accessed by the user that owns the drawer.

Click on the drawer to open it and view the list of reports.

The File View

This view contains the files that are available to be opened by the user. Click on the report description to view the report. The headers in each column are active to provide sorting.

Available Reports	Select?	Job Name	By	Date	Time	% Complete	Pages
User Access Report	☐	REP~17978-10193	Bob Garrard	21/03/17	22:45	100%	4
User Access Report	☐	REP~17978-10192	Bob Garrard	21/03/17	22:30	100%	4
User Access Report	☐	REP~17978-10191	Bob Garrard	21/03/17	22:30	100%	4
User Access Report	☐	REP~17978-10190	Bob Garrard	21/03/17	22:30	100%	4
User Access Report 3	☐	REP~17978-10187	Bob Garrard	21/03/17	22:27	100%	4
User Access Report	☐	REP~17978-10181	Bob Garrard	21/03/17	18:14	100%	4
User Access Report 5	☐	REP~17978-10182	Bob Garrard	21/03/17	18:14	100%	4
User Access Report 6	☐	REP~17978-10183	Bob Garrard	21/03/17	18:15	100%	4
User Access Report 7	☐	REP~17978-10184	Bob Garrard	21/03/17	18:15	100%	4

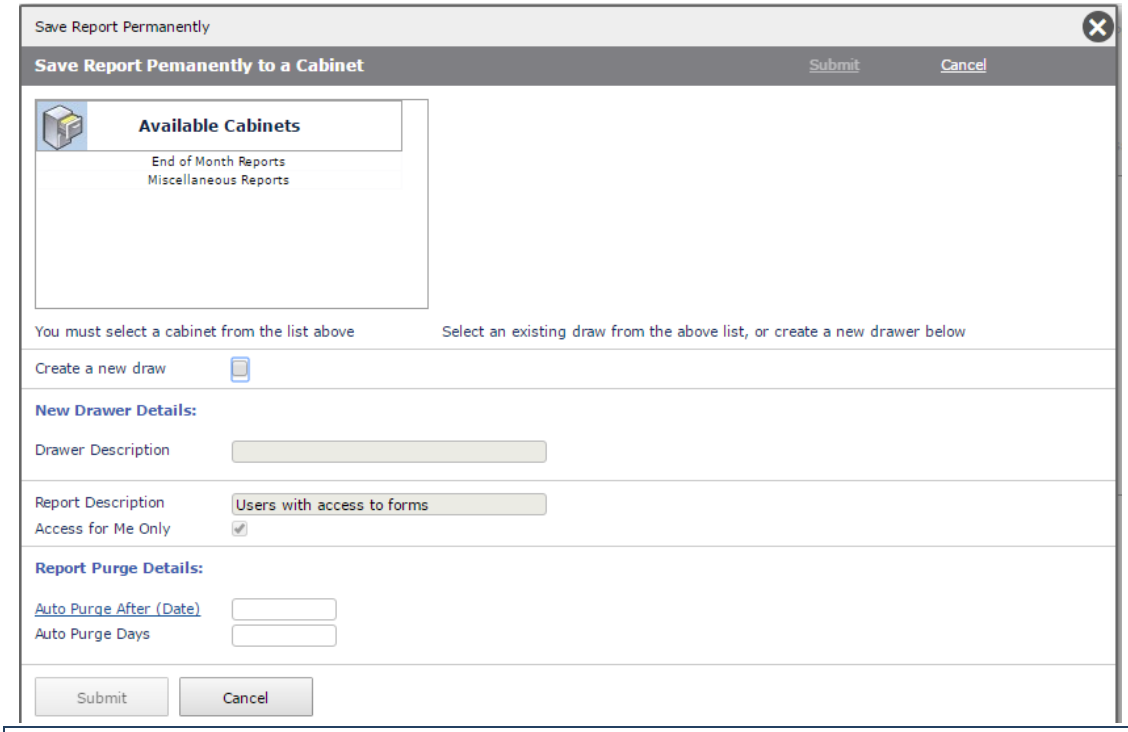
Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Saving a Report to a Cabinet

Every report that is run to preview can be stored into a cabinet.

Every report that is run to preview automatically stores a spreadsheet version. When saved to a cabinet, the spreadhsheet version is also saved.

In report preview mode, there in an option to **Store the report permanently.** When this option is selected, the following form is displayed.



Select a Cabinet

A cabinet must be selected in order to save a report.

Select a Drawer

When a cabinet is selected, a list of available drawers is displayed.

A temporary drawer will automatically be created for every user.

Click on the required drawer to save the report into.

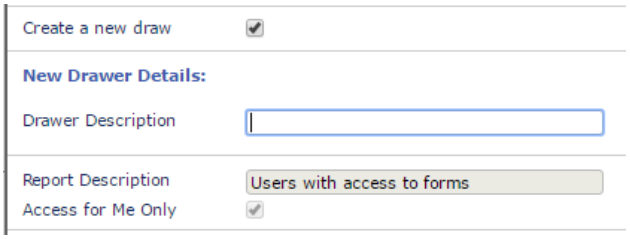
Create a New Drawer

There is an option to create a new drawer. Refer to the screenshot above.

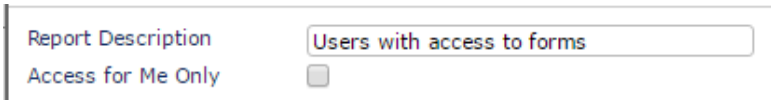
Tick the check-box to indicate that you wish to create a new drawer.

The following prompt will be enabled.

The drawer description can be any text. It is recommended that the description be meaningful to the users that the drawer is intended for.



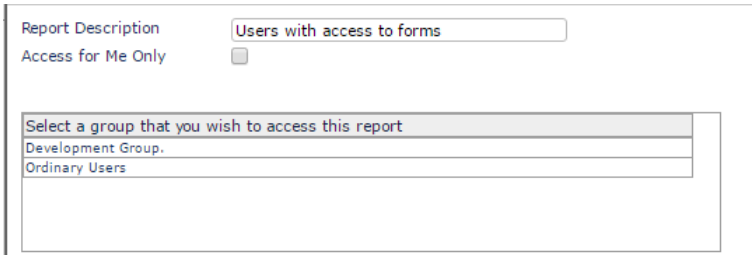
Assign a Report Description



Assigning Access to other users

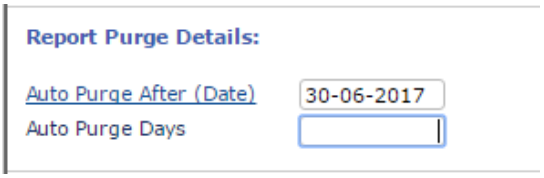
There is an option to provide other users access to a report and a drawer. If the Access for Me Only field is unticked there is list of User Groups displayed that can be assigned access to the report.

If no groups are selected, all users will have access to the report.



Purge Dates

When saving a report, a user can nominate a date to purge [Auto Purge After (Date)] or a number of days (after the save date) to purge the report.



Purging Reports or Moving Reports Between Cabinets

Developers can use **DBI.G.CABINETS** to purge Cabinet drawers or move reports between cabinets.

This subroutine can be invoked as follows:

```
$INCLUDE DBI DBI.COMMON
$INCLUDE DBI DBI.SUB.COMMON
CALL DBI.G.INITVARIABLESNET
CALL DBI.G.OPENNET
*set variables
CABINET       = name of cabinet to purge (or source cabinet for a move)
DRAWER        = name of drawer
CABINET2      = name of cabinet to move reports to (only needed for CAB.ACTION = "MOVE")
DRAWER2       = name of drawer to move reports to
CAB.ACTION    = either "MOVE" or "PURGE"
DELETEEMPTY   = 1 (delete empty drawer) or 0
*
CALL DBI.G.CABINETS(CABINET,DRAWER,CABINET2,DRAWER2,CAB.ACTION,DELETEEMPTY)
RETURN
```

Notes:

- the cabinet and drawer arguments must contain the name NOT the description of the entity
- the purge only occurs when the purge conditions, set when the report is saved, are met

If developers wish to purge Cabinets in the *End of Period* then use the subroutine **DBI.G.CABINETPURGE**.
This routine expects parameters to be passed via the DesignBais common variable PROCESS.PARAMETER.
Parameters must be passed as a comma separated string.

PROCESS.PARAMETER csv position 1: *Name of Cabinet* to purge or from which reports are to be moved

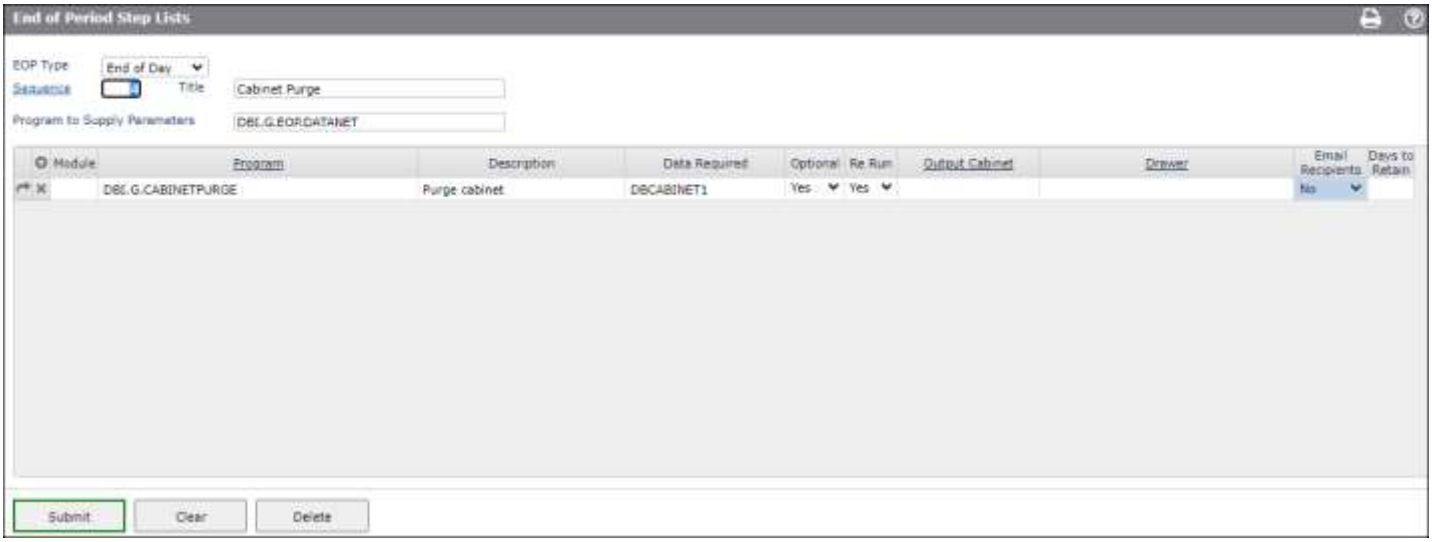
The next 2 postions can be null:

PROCESS.PARAMETER csv position 2: *Action* – either *PURGE* or *MOVE* (null is interpreted as PURGE)
PROCESS.PARAMETER csv position 3: Delete Empty drawer flag – Y or 1 delete empty drawer after purge

If Action is MOVE then:

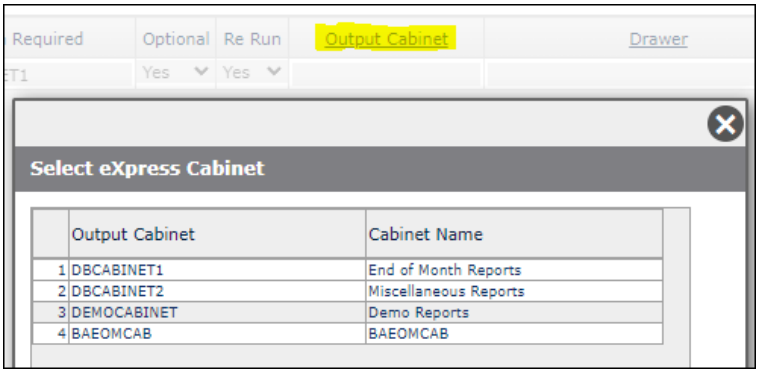
PROCESS.PARAMETER csv position 4: Name of *Drawer* to be moved
PROCESS.PARAMETER csv position 5: Name of *Cabinet* containing the drawer that reports will be moved to
PROCESS.PARAMETER csv position 6: Name of *Drawer* that reports will be moved to

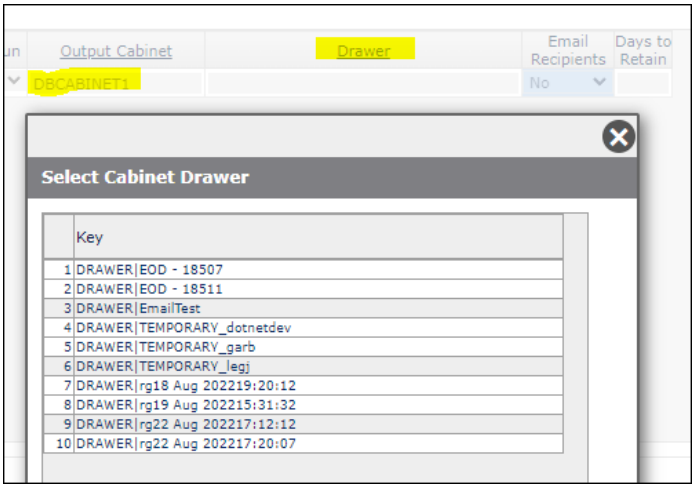
Example of *End of Period* setup:



Note that the Program to Supply Parameters DBI.G.EOP.DATANET is required to load the *Data Required* into PROCESS.PARAMETER at run-time.

From Release 8.8.2.1 there is now a header process lookup for both *Output Cabinet* and *Drawer*.





Running Reports to a Cabinet Remotely

You may run a report to a cabinet from anywhere in your application. This allows the creation of automated back-up/copy of reports produced. It also makes multi-task operations very easy. All report output from multi-task operations can be sent directly to cabinets.

The following is a code example that has all the elements required to submit a report remotely.

```
$INCLUDE DBI DBI.COMMON

PROCESS.PARAMETER    = "REMOTECABINET"           ;* REMOTECABINET is the flag for the report generator
PROCESS.PARAMETER<2> = "DBCABINET2"              ;* Cabinet File. Must be valid File within current account
PROCESS.PARAMETER<3> = "FEBRUARY2007"            ;* Name of the drawer in the cabinet
PROCESS.PARAMETER<4> = "February End of Month"   ;* Label associated with the drawer | May be changed
PROCESS.PARAMETER<5> = "FEB0709"                 ;* File Name. If not unique overwrites file in same drawer
                                                 ;* If Null, job name is returned back to the calling process
PROCESS.PARAMETER<6> = "Ytd Sales by Test5"      ;* Report Description
PROCESS.PARAMETER<7> = "DBCLIENT*REP.AFTER"      ;* Report to run
PROCESS.PARAMETER<8> = "DBI.DEV"                 ;* DBIACCOUNT - for the phantom
PROCESS.PARAMETER<9> = WEBLOGON                  ;* WEBLOGON. User to be given access to the report
                                                 ;* If Null, all users have access
PROCESS.PARAMETER<10> = DBDATEFORMAT             ;* for the phantom
PROCESS.PARAMETER<11> = SESSION.ID               ;* SESSION.ID - for the phantom
PROCESS.PARAMETER<12> = ""                       ;* DB Groups that can access this report (VM Delimited)
PROCESS.PARAMETER<13> = ""                       ;* Cabinet Purge Date - Will be purged after this date
PROCESS.PARAMETER<14> = "30"                     ;* Cabinet Purge Days - Will be purged after a number of days
PROCESS.PARAMETER<15> = WEBSERVER                ;* Webserver path for the link to images

DBUSAGEVAR<11> = "evans"                         ;* Results from a selection for - This is offset by 10

CALL DBI.I.STARTPRINTNET                        ;* not provided.
```

Highchart Templates

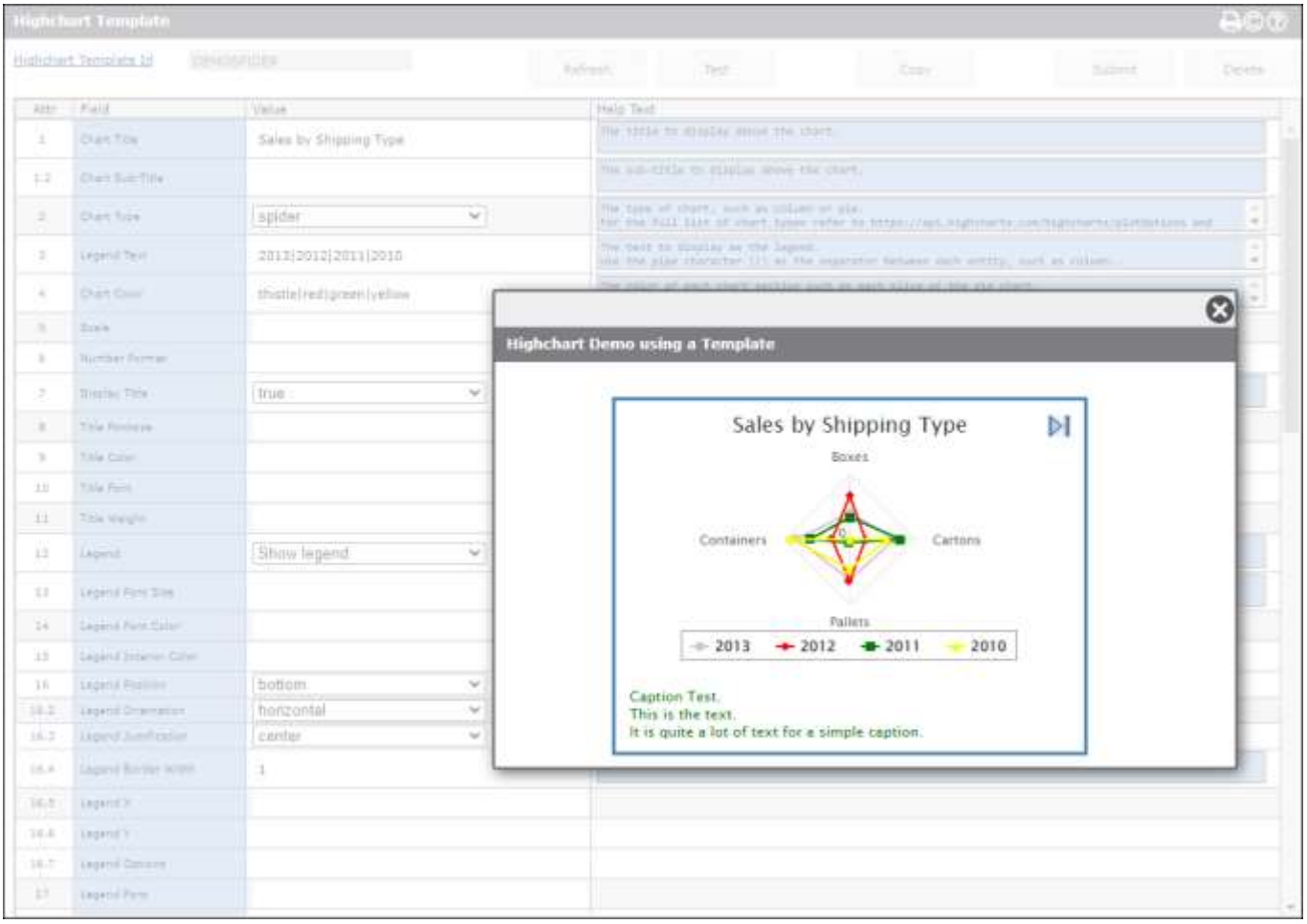
Highcharts can be displayed on a DesignBais form by adding the *Highcharts graph* field type element to the form. The fieldname will default to *H.HICHARTn* where *n* is the next available report number for the form.

The basic code to display the chart is the same as is used for a standard on-form report.

OUTPUT.ATTR is used to load the highchart options. The data for the chart is loaded into *OUTPUT.REPORT*.

Developers can use highchart templates to hold standard chart options and then use *OUTPUT.ATTR* to override selected template values at run time. Assign the string '$BLANK$' (without the quote marks) to an OUTPUT.ATTR attribute in order to set a template value to null.

In the example below the options to produce a spider chart have been entered. The *Test* button allows the developer to see resulting chart using sample generated data.



Chapter 13 – Code Block Usage

Code Block Usage

What is a Code Block?

A Code Block can be attached to the majority of subroutine calls throughout DesignBais. In DesignBais Releases pre version 4.13, all code needed to be in a program file. This meant that for even the simplest task, a program was required.

A Code Block is still a program. It resides in a file named DBICODEBLOCK. You will need to ensure that your account has a DBICODEBLOCK file and the appropriate object file if your database requires it.

The Code Block links itself to a Field Property (Dictionary) or a field on a form or a report.

Code Block Example

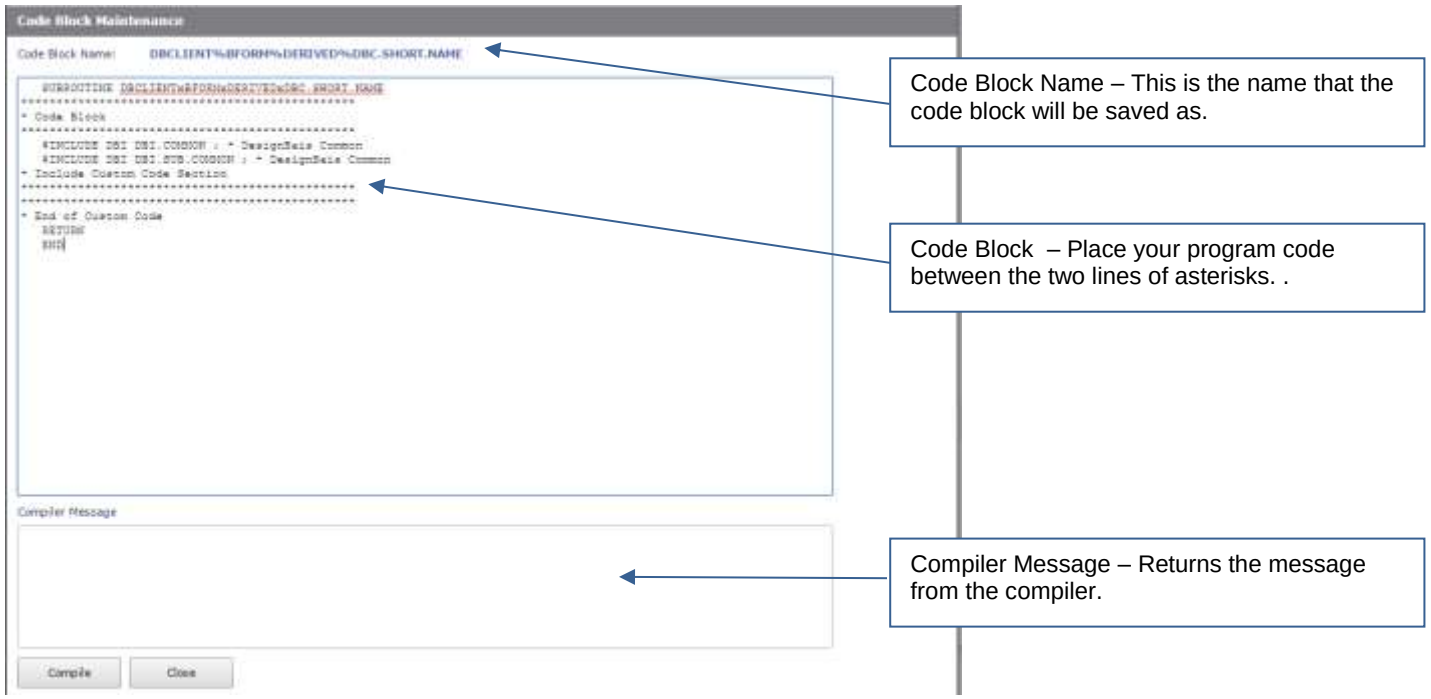
Code Blocks are ideal for derived fields.

Code Blocks can be used wherever you would normally call a BASIC program. They are particularly useful when calculating derived fields on a report.

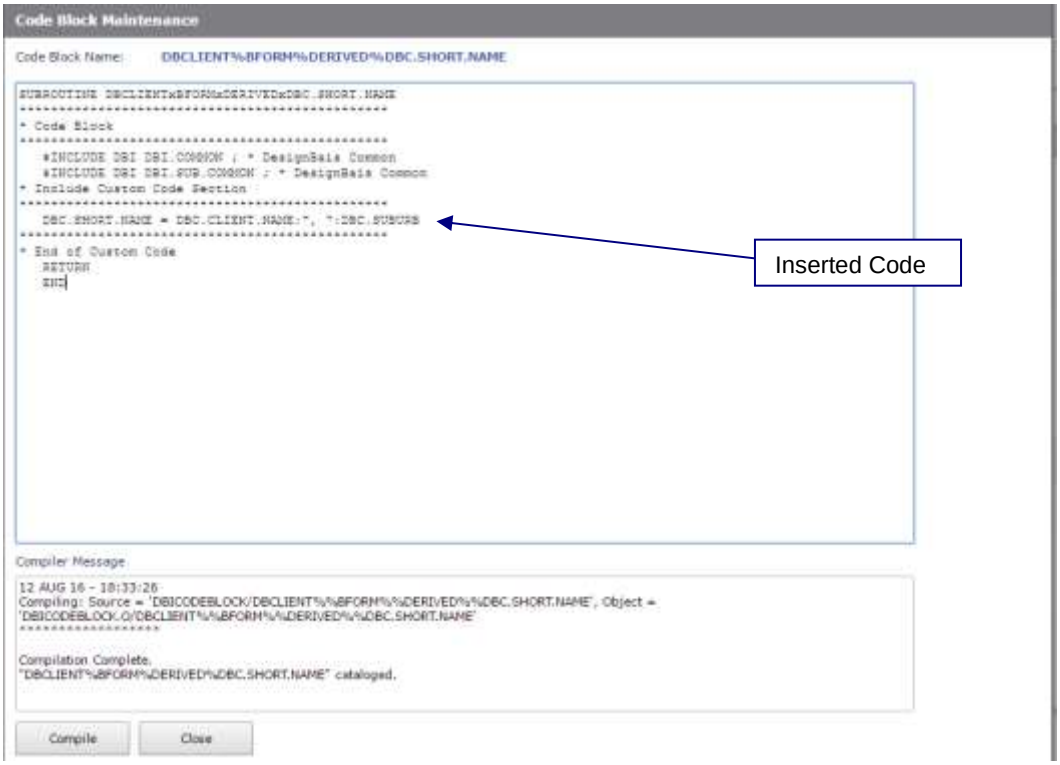
In this example there is a Code Block attached to the “Derived in Subroutine” field for the Field DBC.SHORT.NAME

This is achieved by entering a “C:” at the prompt. You will notice that the View Code Block and Derived Field Parents controls are displayed at this point.

Click the View Code Block hyperlink to view the Code Block form.
There are three main sections in Code Block Maintenance.



Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3



Example Code Block Code

In the example above, the code: **DBC.SHORT.NAME = DBC.CLIENT.NAME:", ":DBC.SUBURB** was all that was needed to derive the field DBC.SHORT.NAME.

DBC.SHORT.NAME is a field on the file DBCLIENT. It has been assigned an attribute of 36.
DBC.CLIENT.NAME is a field on the file DBCLIENT. It has been assigned an attribute of 1
DBC.SUBURB is a field on the file DBCLIENT. It has been assigned an attribute of 18.

The code block pre-compiler has established that all of the field names entered in the program were Field Properties in the DesignBais file named DBIPROP.

This then allows the developer to construct simple expressions using the names of field properties as apposed to the allotted field attributes.

The main advantage of this is that is makes code block portable. A code block assigned at the Field Property level, can be placed on any form. DesignBais will establish what DBRECORD or DBOTHER.RECORD the field names belong to at run time.

If the above code was programmed in non Code Block mode, it would be as follows:

DBRECORD<36> = DBRECORD<1>:", ":DBRECORD<18>

This is still correct and can be used in code blocks.

Code Block Naming Convention

In the above example, the code block name is: DBCLIENT%BFORM%DERIVED%DBC.SHORT.NAME
and the internal name of the subroutine is: DBCLIENTxBFORMxDERIVEDxDBC.SHORT.NAME (for cross-platform compatibility)

The Code Block name can be split into four components

Filename: **DBCLIENT**
Form Name: **BFORM**
The form name is blank for code blocks placed against field properties.
Event Type: **DERIVED**
The event type can be any of the DesignBais Event Types. Eg. VALIDATE, AFTER.DISPLAY, MV.POST
Where an event type normally has a space, it is replaced by a “.” (dot) for code block naming.
Field Name: **DBC.SHORT.NAME**
Name of the Field being represented by the code block.

There are two records written to the file DBICODEBLOCK.

1. The original source code. This is written as *Code Block Name.SRC*

Eg. DBCLIENT%BFORM%DERIVED%DBC.SHORT.NAME**.SRC**

```
SUBROUTINE DBCLIENTxBFORMxDERIVEDxDBC.SHORT.NAME
*********************************************
* Code Block
*********************************************
   $INCLUDE DBI DBI.COMMON ; * DesignBais Common
   $INCLUDE DBI DBI.SUB.COMMON ; * DesignBais Common
* Include Custom Code Section
*********************************************
   DBC.SHORT.NAME = DBC.CLIENT.NAME:", ":DBC.SUBURB
*********************************************
* End of Custom Code
   RETURN
   END
```

2. The actual code that was compiled. This is the pre-compiled version of the original source

Eg. DBCLIENT%BFORM%DERIVED%DBC.SHORT.NAME

```
SUBROUTINE DBCLIENT%BFORM%DERIVED%DBC.SHORT.NAME
*********************************************
* Code Block
*********************************************
   $INCLUDE DBI DBI.COMMON ; * DesignBais Common
   $INCLUDE DBI DBI.SUB.COMMON ; * DesignBais Common
   DBX.TOFIND = "DBC.CLIENT.NAME,DBC.SUBURB,DBC.SHORT.NAME"; CALL DBI.G.TOFIND
* Include Custom Code Section
*********************************************
   DBX.RTNV<3> = DBX.RTNV<1>:", ":DBX.RTNV<2>
*********************************************
* End of Custom Code
   RETURN
   END
```

This version of the Code Block should never be modified manually.

Chapter 14 – Common Variable usage and Subroutine Interaction

Common Variable usage and Subroutine Interaction

This section of the manual is dedicated to the use of common variables and how these variables affect the way the client application interacts with the database server.

These common variables are within the supplied DBI.COMMON item, which resides in the DBI file. Only the variables that are detailed in this document should be used by the developer. Non-documented variables are for DesignBais internal use only.

Every new browser window creates a unique set of session variables. The common variables associated with each session are detailed in this section.

Storage Variables

A 'Storage Variable' is used to maintain records within a user session.

These variables may be referred to or modified by the developer.

DBRECORD

DBRECORD is the main variable for data storage on screen forms. It is a single dimensioned dynamic array.

Typically, DBRECORD is used as the main data container for most DesignBais forms or form-sets.

DBRECORD corresponds with the key variable DBKEY.

DesignBais will maintain DBRECORD throughout a user's interaction with a form or form-set. Developers can refer to and change values within BASIC subroutines.

DBRECORD is also used as the storage variable within the DesignBais report generator. DBRECORD is assigned for each item read from a select list in a report.

DBORIGINAL.RECORD

Contains a copy of DBRECORD as it was originally read.

DBOTHER.RECORD*(n)*

DBOTHER.RECORD is a dimensioned array with 99 usable elements. Element 100 is preserved for DesignBais internal usage.

Typically DBOTHER.RECORD is used to store details of supporting records on a DesignBais form or report.
This allows the developer to read and write up to 100 different files/records within a single form or form-set.

DesignBais will maintain DBOTHER.RECORD throughout a user's interaction with a form or form-set. Developers can refer to and change values within BASIC subroutines.

DBORIGINAL.OTHER.RECORD*(n)*

Contains a copy of DBOTHER.RECORD*(n)* as it was originally read.

DBWORK

DBWORK is primarily designed to be a working variable.

DBWORK should be used for input fields that construct multi-part keys.

DesignBais will maintain DBWORK throughout a user's interaction with a form or form-set. Developers can refer to and change values within BASIC subroutines.

DBKEY

DBKEY contains the key associated with the read and updates of the DBRECORD variable.
DBKEY can be modified or assigned in the “BEFORE WRITE” event if the developer wants to update a record with a new key.

If the record being written is a new key then DBKEY must be used in conjunction with DBRECORD.STATUS to bypass the DesignBais optimistic locking requirements.

```
Eg.   IF DBWORK<KEY.STATUS.WK> = “NEW” THEN
        DBRECORD.STATUS = 1 ; * Set to 1 to bypass optimistic check
        DBKEY = New Key Value
      END
```

DBKEYS(*n*)

DBKEYS(*n*) contains the values of the keys associated with DBOTHER.RECORD(*n*).

DBKEYS have 99 usable subscripted elements.

DBKEYS can be modified or assigned in the “BEFORE WRITE” event if the developer wants to update a record with a new key. If the record being written is a new key then DBKEYS must be used in conjunction with DBOTHER.RECORD.STATUS to bypass the DesignBais optimistic locking requirements.

DBVALUE

DBVALUE contains the value of the last input field that invoked a server event when changed. Developers may refer to and modify the value of this variable in BASIC subroutines. This provides for a common method of validating input within BASIC subroutines.

The value in DBVALUE does not get assigned to DBRECORD, DBOTHER.RECORD or DBWORK until validation is successful.

DBUSAGEVAR

DBUSAGEVAR is used for selection processes only. DBUSAGEVAR contains some internal DesignBais data. It also contains the results from the input fields on a search form. The position of the data is offset by ten attributes.

Eg. In a search form with two input fields, Client Name and Town/City, the user input from this form would be stored with Client Name in DBUSAGEVAR<11> and Town/City in DBUSAGEVAR<12>

KEY.PART(*n*)

KEY.PART(*n*) stores the individual key elements of every record read on a form or form-set.

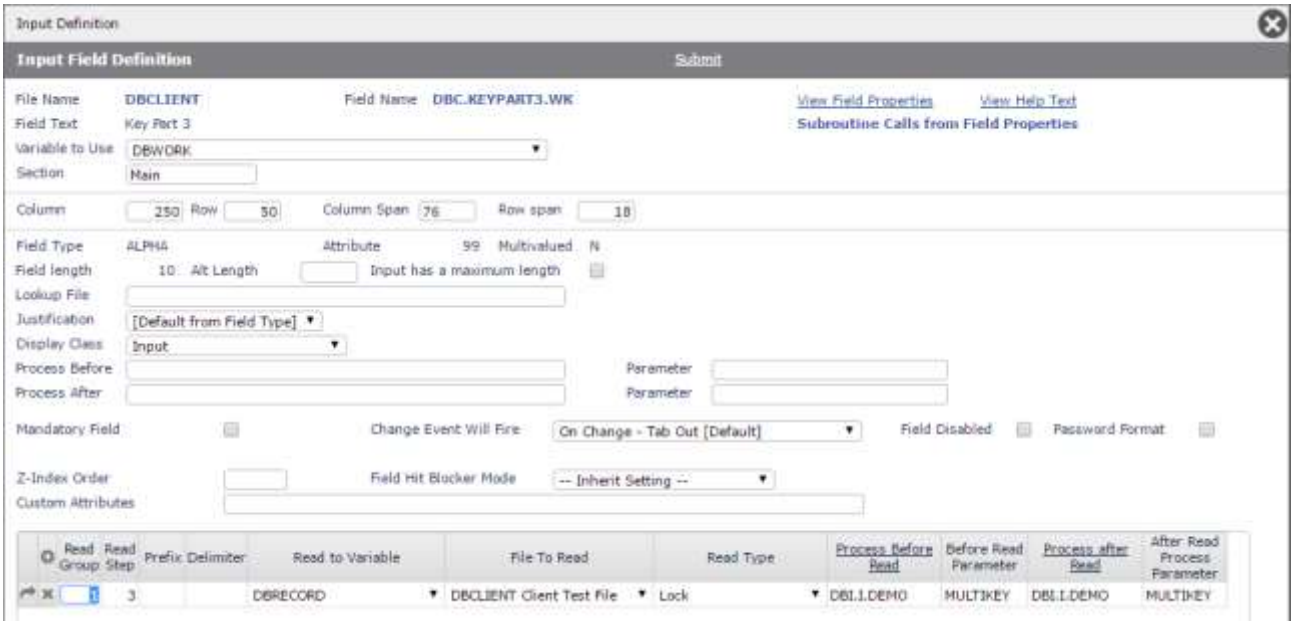
The subscript used is in direct relationship with the Read Group used for each read. Each Read Step is stored as a value within the subscript aligned to the read group.

```
Eg.   Multi-part key containing Order Number and Company
      Read from MYORDERS file against read group 1.

      There are two read steps.
      Read step 1 is against an input field for Order Number stored in DBWORK<1>
      Read step 2 is against an input field for Company store in DBWORK<2>

      Values entered are ON1234 and CO1

      KEY.PART(1)<1,1> contains ON1234
      KEY.PART(1)<1,2> contains CO1
```

Developers should never modify the contents of KEY.PART, but only use it as a reference if required. In the above example, the developer could change the KEY.PART references in a BASIC subroutine by changing the values within DBWORK<1> or DBWORK<2>. The exception to this is if the developer wants to completely clear the state of all form variables. In addition to clearing DBRECORD, DBWORK, etc, use MAT KEY.PART = “” or a similar platform-specific command to clear all key parts which are currently being tracked.

When working with multi-part keys in the form subroutine note that any of the DBWORK fields in the read can appear in the PROCESS.EVENTSOURCE depending on which key part has changed and thus forced the read to occur. The EVENTSOURCE will correspond to the name of the field that changed. The initial read will occur when all key parts have been entered and therefore the EVENTSOURCE will contain the field name of the final key part field. But if all key parts are populated and the first key part is changed then EVENTSOURCE will contain the name of the first key part field.

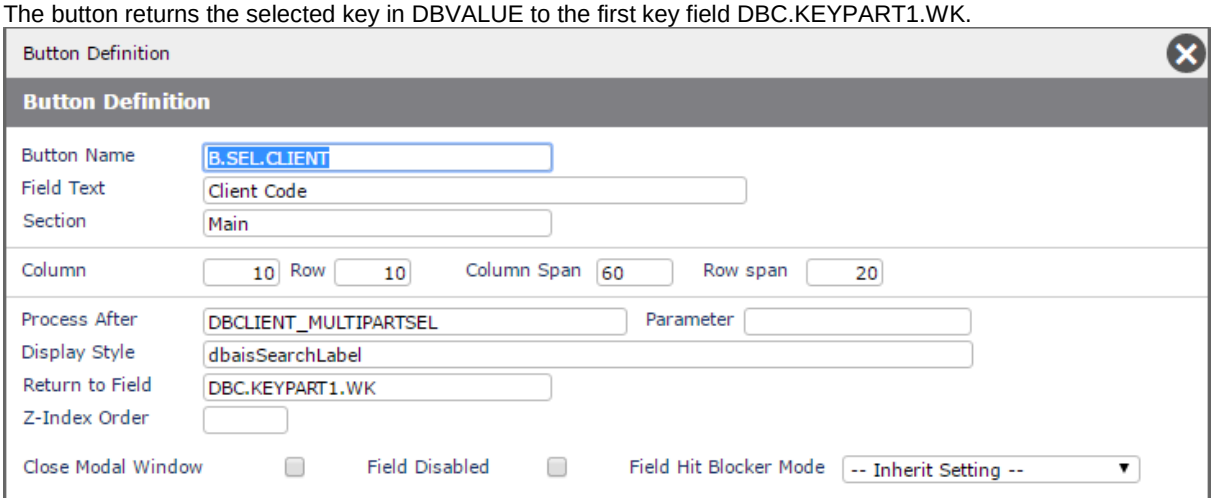
For this reason it is often easier to use the PROCESS.PARAMETER value within the form subroutine. This is shown in the following example where the BEFORE READ uses field names while the AFTER READ uses PROCESS.PARAMETER.




```
*
BEFORE.READ:
*
    BEGIN CASE
      CASE EVENTSOURCE = "DBC.KEYPART3.WK" AND SCREEN.NO = "MULTIKEY"
        * Note that multi-part key processing requires all fields in the read to be intercepted
        * since EVENTSOURCE will correspond to the part of the key that changed and thus forced
        * the read to take place
        NULL
      CASE EVENTSOURCE = "DBC.KEYPART2.WK" AND SCREEN.NO = "MULTIKEY"
        NULL
      CASE EVENTSOURCE = "DBC.KEYPART1.WK" AND SCREEN.NO = "MULTIKEY"
        NULL

    END CASE
    RETURN

*
AFTER.READ:
*
    BEGIN CASE
      CASE SCREEN.NO = "MULTIKEY" AND THIS.PARAMETER = 'MULTIKEY'
        * since EVENTSOURCE will correspond to the part of the key that changed and thus forced
        * the read to take place it is easier to set THIS.PARAMETER and check for this
        NULL

    END CASE
    RETURN
```

The following demonstrates how to populate the multi-part key fields when using a selection process to select an existing record. In the example below button B.SEL.CLIENT runs a selection process to allow the user to select a record. The key of the selected record will be of the form KEYPART1*KEYPART2*KEYPART3.

The button returns the selected key in DBVALUE to the first key field DBC.KEYPART1.WK.



The form subroutine then extracts the 3 elements of the key and populates the 3 DBWORK fields. Finally DBVALUE is changed to contain just the value required by the first key part field.

```
*
AFTER.SELECT:
*
      BEGIN CASE
         CASE EVENTSOURCE = 'DBCLIENT_MULTIPARTSEL'
            DBWORK<DBC.KEYPART1.WK> = FIELD(DBVALUE,'*',1)
            DBWORK<DBC.KEYPART2.WK> = FIELD(DBVALUE,'*',2)
            DBWORK<DBC.KEYPART3.WK> = FIELD(DBVALUE,'*',3)
            DBVALUE = FIELD(DBVALUE,'*',1)
      END CASE
      RETURN
```

Note that key field validations occur as part of the read. If there is a validation process on the 2nd part of a two-part key, and the 2nd key is entered before the 1st part of the key then the validation event will not happen.

To get around this use section control or other means to disable the 2nd key part until the 1st key part has been entered.

DBENQUIRY.MODE

Indicates that a form is displayed in enquiry mode. Input fields are disabled. DBENQUIRY.MODE turns multivalue Input fields into output fields.

The field state is flagged in MVOUTREC<5> where the following values indicate the mode:

0 = Input
1 = readonly
2 = output
3 = disabled

The <td> cells containing <input> fields are given an ID = ASSOCzDBWLEVELzROWzCOL while the <input> element gets a full ID like ASSOCz*fieldType*vDBWLEVELzROWzCOL. In enquiry mode the full ID is applied to the <td>.

KEY.DELIM(*n*)

KEY.DELIM(*n*) contains the delimiters used to construct a multi-part key. Each KEY.DELIM subscript corresponds with the matching KEY.PART subscript. Developers should not modify the values of KEY.DELIM programmatically.

DBSTORE(*n*)

Is a multi-dimensioned array with 200 subscripts. Developers can store data (including complete records) in each subscript. DesignBais will maintain the values in DBSTORE for the entire user session.

DBRECORD.STATUS

Determines whether the DBRECORD is a new record or not.
Properties

0	Indicates that the record was read.
1	Indicates a new record
2	Indicates that the record cannot be retried due to an existing record lock

If the record being written is a new key then DBKEY must be used in conjunction with DBRECORD.STATUS to bypass the DesignBais optimistic locking requirements.

```
Eg:     IF DBWORK<KEY.STATUS.WK> = “NEW” THEN
          DBRECORD.STATUS = 1 ; * Set to 1 to bypass optimistic check
          DBKEY = New Key Value
        END
```

DBOTHER.RECORD.STATUS

Determines whether the DBOTHER.RECORD(*n*) is a new record or not.

DBOTHER.RECORD.STATUS is value mark delimited for each of the DBOTHER.RECORD reads.

Properties

0	Indicates that the record was read.
1	Indicates a new record
2	Indicates that the record cannot be retried due to an existing record lock

If the record being written is a new key then DBKEYS must be used in conjunction with DBOTHER.RECORD.STATUS to bypass the DesignBais optimistic locking requirements.

```
Eg.     IF DBWORK<KEY.STATUS.WK> = “NEW” THEN
          DBOTHER.RECORD.STATUS<1,5> = 1 ; * Set to 1 to bypass optimistic check for DBOTHER.RECORD(5)
          DBKEYS(5) = New Key Value
        END
```

Error Dialog and Display Dialog Variables

There are two common variables used to display warning and message boxes.

IERR.TEXT

IERR.TEXT is used to display an error message dialog that contains the string assigned to the variable.

Usage: IERR.TEXT = *"Error Text"*

Eg. IERR.TEXT = "Not a valid entry"

DesignBais checks for a non-null state for IERR.TEXT. If found the following occurs.

1. A dialog containing the message is displayed.
2. If set in a validation event the original value of the field is retained and DBRECORD, DBOTHER.RECORD or DBWORK is not updated with the entered value.

If IERR.TEXT is not null then all actions that the developer may have defined in Basic code will not take place. The exception is that assigning a field name to DBRETURN.TO.FIELD will still cause focus to move to the designated field after the error message has displayed.

A call to DBI.G.GLOSSARY is used to build glossary entries from error messages.

DBDS (DesignBais Display/Debug String).

DBDS is used to display a message dialog that contains the contents of the string assigned to the variable.

This is a very useful debugging tool for developers.

Usage: DBDS<-1> = *"String to be displayed"*

Eg. Display a message dialog for debugging purposes.

```
CASE PROCESS.EVENTSOURCE = "DBCLIENT_SEARCH1"
   DBDS<-1> = "DBUSAGEVAR<11> = ":DBUSAGEVAR<11>
   DBDS<-1> = "DBUSAGEVAR<12> = ":DBUSAGEVAR<12>
   DBDS<-1> = "DBUSAGEVAR<13> = ":DBUSAGEVAR<13>
   DBDS<-1> = "DBUSAGEVAR<14> = ":DBUSAGEVAR<14>
```

Custom dialogs are created with a call to DBI.G.DIALOG.

Session Variables

There are a number of variables associated with the unique browser session. These are used to help the developer identify the user, the account or directory, the server and the session identifier for each browser session.

DBCOOKIE

Is a unique key associated with the client computer. DBCOOKIE does have the browser ID but a browser may have several tabs which are identified by the SESSION.ID.

DBIACCOUNT

Is the name of the database account or the current database directory. Changing this value within a subroutine will force DesignBais to change the current account for the user. .

DBDEBUG

If true (1) the subroutine was invoked at TCL by DBI.RUN.LAST.

DBGLOSSARY

Is the value of the current glossary. Changing this within a subroutine will change the current glossary being used by a user. Ensure that you set DBGLOSSARY<1,1> to the new glossary code rather than merely setting DBGLOSSARY itself. DBGLOSSARY<1,2> through <1,4> is used for glossary type, glossary source text and glossary action respectively.

DBLOGGING

If true (1) user logging is currently turned on. .

DBSCREENPROPS

Contains the screen size. Pipe delimited depth and width. This is assigned at the get session stage and is not refreshed again throughout the session. It does not change if the window is resized.

```
WINDOW.WIDTH    = FIELD(DBSCREENPROPS,"|",1)
WINDOW.DEPTH    = FIELD(DBSCREENPROPS,"|",2)
MAX.COL         = FIELD(DBSCREENPROPS,"|",3)
MAX.ROW         = FIELD(DBSCREENPROPS,"|",4)
```

DBW3C

DesignBais always operates in W3C mode hence this variable is set to 3.

DBW3CACTION

Contains the qcode value from the url. For example from *ac=legj* the variable DBW3CACTION='legj'. DBW3CACTION is null after the log in form DBIGLOBAL_D20.

DBW3CBROWSERVERSION

Contains details of the browser version.

DBW3CTITLE

Is used to define the title of the browser window.

DBWLEVEL

Records the level of the current form. It is 1 at the base level and increments as each higher level is executed. Calling a modal form or calling a form with '~M', '~S', '~L' or '~LM' appended to the form name will increment the value in DBWLEVEL. The value in DBWLEVEL must never be amended by the developer.

For the developer the following variables may be useful. Do not amend the value in these variables.
If DBWLEVEL > 1 then DesignBais is not processing a base form. The form must be modal or layered.
DBLAYER.FORMS contains a MV list of layered forms.

DBFORM.STACK contains a MV list of forms.
DBLAYER.VALUE contains the MV position in DBLAYER.FORMS. So a value > 0 implies that a layered form is being processed.

DBSOURCEIP

DBSOURCEIP Contains the source IP address of the client machine.

SCREENROOT

The name of the current form or report. This has the format of ***Filename_Formname.*** In designer mode the form name will have '.DESIGNER' appended to the form name.

ORIGINAL.SCREENROOT

When calling a form there are a number of extra parameters that can be used to affect the mode of the form when loaded. An example may be the parameter "~M" which will load the form as modal. This variable contains the entire string that was used to load the form, where SCREENROOT only contains the actual formname.

Eg. PROCESS.STACK = "DBCLIENT_FORM1~M~E" Load form as modal in enquiry mode.
SCREENROOT will have the value 'DBCLIENT_FORM1'
ORIGINAL.SCREENROOT will have the value 'DBCLIENT_FORM1~M~E'

This allows the developer to check the state of the form programmatically. Note that the value of ORIGINAL.SCREENROOT must be checked in the AFTER DISPLAY event since it will not retain the extra parameters after this point. Refer to the section **Calling Forms** for further detail.

SESSION.ID

The SESSION.ID is a unique number that is used as a storage key for all activities associated with a browser session.

This can be used as an alternative to the telnet port number. All session related data is stored in the file DBISESSIONS.

WEBLOGON

Is the user identifier for the unique session. This is usually associated with the username entered as part of the user authentication process. This value is used to replace the @WEBLOGON tag in forms and reports.

WEBSERVER

This field contains the full URL that was used to invoke DesignBais.

DBSUSPEND.AFTER.READ

This common variable, initialised in DBI.G.INITVARIABLESNET, is used in BAWEBEXECNET during reads to bypass the "VALIDATE" and "AFTER READ" events.

An example of where Developers may need to use this feature is where a read fails and it is necessary to by-pass the VALIDATE and AFTER READ processing for all ensuing reads that are triggered by the failed read.

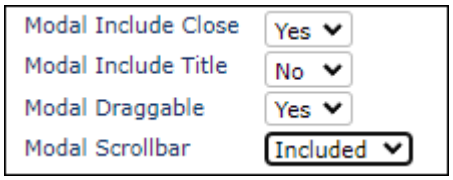
DBHIDECLOSE

Use this variable to hide the close button and suppress the title on a modal form.
DBHIDECLOSE<1> = 1 or Y - hide the close button. Any other non-null value will force display of the close button.
DBHIDECLOSE<2> = 1 or Y - hide the title text on the modal form. Any non-null value will display the title.

From Release 8.9.1.1 this variable has been extended:
DBHIDECLOSE<3> = 1 or Y - prevents drag. Any other non-null value will allow the modal to be dragged.
DBHIDECLOSE<4> = 1 or Y - sets no scroll. Any other non-null value will allow the modal to scroll.

The setting remains until reset by the developer.

Note that these actions can be controlled by the style group settings:



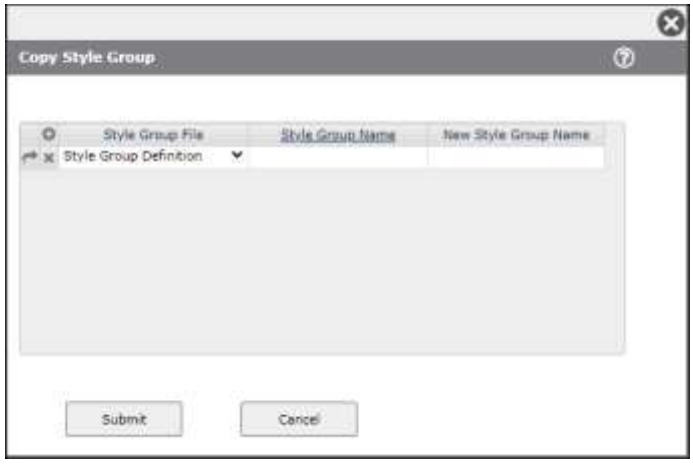
DBHIDECLOSE overrides the style group settings.

The effect of setting:

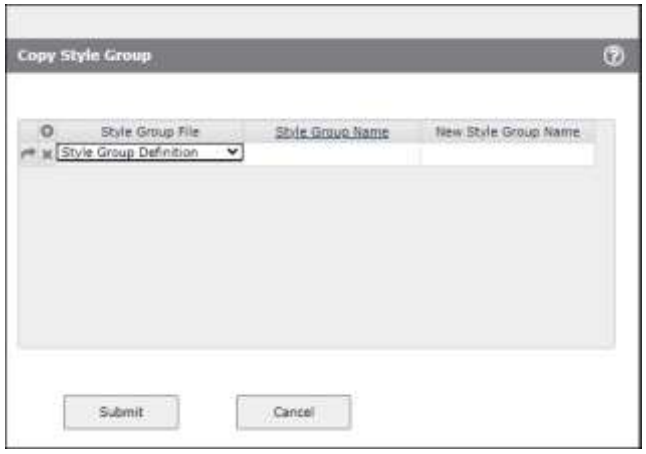
DBHIDECLOSE<1> = 1

DBHIDECLOSE<4> = 1

can be seen in these before and after images.



The form close X no longer displays. The wide right hand border has been reduced by removing the allowance for a scroll bar.



DBMODALPOS

The new common variable DBMODALPOS carries information about the position of a modal form.

DBMODALPOS<1,DBWLEVEL,1> - true / false has been dragged
DBMODALPOS<1,DBWLEVEL,2> - top position – set from dragged or in DBI.G.FOOTERMNET
DBMODALPOS<1,DBWLEVEL,3> - left position
DBMODALPOS<1,DBWLEVEL,4> - level nn as extracted from dbBackGroundnn (element name) used to set the correct MV position
DBMODALPOS<1,DBWLEVEL,5> - Last known height – set in DBI.G.FOOTERMNET

When a modal is closed or opened DBMODALPOS<1,DBWLEVEL> will be cleared.

The modal form height is recalculated on each hit, in subroutine DBI.G.FOOTERMNET, in case collapsed sections have changed. The modal top position will only be amended when all the following criteria are met:

- the height has increased
- it is not a new modal opening
- the application is not adjusting the top position via DBAJAXCMD. If the developer has used either a call to DBI.G.AJXCMD, or has set DBAJAXCMD<-1>, then the position is derived from the DBAJAXCMD string.
- the form hasn't been dragged or the last known top + the new height takes the bottom of the form out of view.

DBI.G.FOOTERMNET sets:

- DBMODALPOS<1,DBWLEVEL,2> = THIS.TOP
- DBMODALPOS<1,DBWLEVEL,5> = THIS.HEIGHT

If dragged=true the top may be overridden.

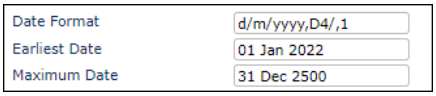
Example:

```
DBMODALPOS<1,2> = 'trueü83.5ü190üdbBackGround2ü290'
```

<1,2,1> the value "true" indicates that the modal form has been dragged
<1,2,2> the top of the form is 83.5 px from the top margin
<1,2,3> the left edge of the form is 190 px from the left margin
<1,2,4> dbBackGround2
<1,2,5> the last known height of the form is 290 px.

DBDATEFORMAT

The date format is set in General Global Parameters. Refer to GeneralGlobalParameters



DBDATEFORMAT<1,1> = d/m/yyyy
DBDATEFORMAT<1,2> = D4/
DBDATEFORMAT<1,3> = 1
DBDATEFORMAT<1,4> = 2022-01-01 (min as yyyy-mm-dd)
DBDATEFORMAT<1,5> = 2500-12-31 (max as yyyy-mm-dd)

This variable should not be changed in application code.

Loading a form within a subroutine

As well as being able to load a form from a menu or as an after event on a button or field, the DesignBais developer can invoke a form load from a BASIC subroutine.

PROCESS.STACK

Used to invoke a form from a program. Only one form can be stacked at any one time.

There are two parameter settings that affect the form load process. These parameters are separated by a tilde ('~')

The **first parameter** controls the style of the form and the environment in which the form operates.

There are five possible values for this parameter.

M **Modal Form**

The form will be loaded as a modal form. If the form was not set-up as a modal form during form design, the type is switched at runtime to modal

S **Standard Form**

The form will be loaded as a standard form and replace the base form on the current layer.

L **Layered Form**

The layered form will be displayed as a modal form. All session variables that pertain to records, work variables, counters etc will be set to null at the time a layered form is loaded. When the layer is closed all common variables at the previous layer are re-instated. Calling a form using PROCESS.STACK with ~L appended will override the called form's Preserve Common and Subform flags.

This 'layering' technique allows the developer to have multiple forms open that all use similar or the same read variables and read groups without modifying the common space associated with other open forms.

LM **Layered Form as Modal**

Opening a form which has the Preserve Common and Sub-Form flags set using the "~LM" option will not clear COMMON as the form is opened but will not return COMMON to the calling form. The calling form therefore will retain the same values in COMMON on return as before the call.

Blank **(no value)**

Will use the design time parameters associated with the form.

The **second parameter** is used to indicate whether the form is run in enquiry mode.

E **Enquiry Mode**

This will switch the loaded form (and subsequent sub-forms) into enquiry mode. No DesignBais related write processing will occur when a form is in enquiry mode.

From within a subroutine the developer can check the status of enquiry mode by referencing the variable DBENQUIRY.MODE. If true (1) enquiry mode is set.

Blank **(no value)**

The form is loaded in update mode.

Usage: PROCESS.STACK = *"Filename_Formname[~Style~Enquiry Mode]"*
Eg. PROCESS.STACK = "DBCLIENT_LESSON1"
The form DBCLIENT_LESSON1 will be invoked.

Eg. PROCESS.STACK = "DBCLIENT_LESSON1~L"

The form DBCLIENT_LESSON1 will be invoked as a modal, layered form.

Eg. PROCESS.STACK = “DBCLIENT_LESSON1~M”
The form DBCLIENT_LESSON1 will be invoked as a modal, non-layered form. The “M” will override the design time modal/non modal property of the form.

Eg. PROCESS.STACK = “DBCLIENT_LESSON1~L~E”
The form DBCLIENT_LESSON1 will be invoked as a modal, layered form in inquiry mode.

It is possible to call a form and and set DBENQUIRY.MODE = 1 in the AFTER DISPLAY event in order to display the form in enquiry mode. Note however that you need to specify on the form which buttons and images be disabled. The submit button will be disabled automatically but other buttons need to be handled by the developer.

are to

Refer to the section **Calling Forms** for further detail.

Using PROCESS.STACK during a READ event can be invalid. In general a read is designed to populate data fields on the current page.

If you want to move back to a previous page during a READ then use DBBUTTONCLICK. This sends a javascript command to the browser meaning that the READ event completes and then you get a click event.

Using a PROCESS.STACK in a validation during a READ will not work.

A value can be passed to PROCESS.PARAMETER by placing a third tilde (‘~’) in the string, followed by the value to which PROCESS.PARAMETER is to be set.

Eg. PROCESS.STACK = “DBCLIENT_LESSON1~L~E~NEXT”

The form DBCLIENT_LESSON1 will be invoked as a modal, layered form in inquiry mode and PROCESS.PARAMETER will be set to a value of “NEXT”.

Calling Forms

The developer can call forms in a number of ways:

- From a menu
- From a Process Slot on a form
- From a Basic Subroutine as described above

In all cases the form can be called as a modal form, a layered form, or an enquiry form (in combination with modal or layered).

Calling a form with an explicit style using the ~M or ~L suffix will override the Modal/Non Modal property that is set by Forms Designer. Calling a form using ~S (Suppress) gives the developer the ability to process stack a form that is defined as modal but force it to run from inside current modal layer, not a new one.

When the developer wants Common to be preserved then the Forms Designer properties Preserve Common and Sub Form should be checked. The form can then be called by using the FILENAME_FORMNAME~M option. This means that DBWORK, for example, will be passed into the called form, and if the called form modifies DBWORK the modified DBWORK fields are passed back to the calling form.

A Sub Form may display a field from the main calling form that has reads associated with it on the main form. In line with the concept of preserving common the Sub Form will not normally have those reads unless the developer is specifically allowing for the field to be amended on the Sub Form. The amended value would then be passed back to the main form. In the case of a Key Field it would be wrong to allow this field to be amended on a Sub Form. The Key Field on the Sub Form should not be an input field.

Where the developer requires that common variables in the calling form are to remain unaffected by any processing done in the called form it is recommended that the developer use the ~L option when calling the form. The called form will still display like a modal form. The size of the modal form (modal window) will be the size indicated by the Form Width and Form Depth set in Forms Designer.

The point being clarified here is that it is often the case that a developer will call a 'modal' form using the ~M style and not want any changes made in the modal form to be passed back to the calling form via common. It is better to call this 'modal' form using the ~L or ~LM option as this will insulate the calling form from any changes to common made in the called form.

Calling a form that has the properties Preserve Common and Sub Form checked (in Forms Designer), using the ~L style, will cause these properties to be ignored. The form will behave as if those properties were not checked.

After Release 8.1.1.6 DBWORK is reset (cleared) when a form (whether a sub-form or not) is called using ~L. Applications that relied on DBWORK being populated and passed into a form called via ~L will need to be modified to pass DBWORK using DBSTORE, or use the ~LM option.

Controlling cursor behaviour

The developer can control cursor behaviour with the use of the following common variable.

DBRETURN.TO.FIELD

This is used to set focus on a field or a button programmatically. The developer can assign a field name or button name to this variable within a subroutine. If the field or button exists focus will be returned to the field or button indicated.

Usage: DBRETURN.TO.FIELD = *"Field name"*

Eg. DBRETURN.TO.FIELD = "DBC.CLIENT.CODE" ; * Will set focus to the client code field

DBRETURN.TO.FIELD = "B.SUBMIT" ; * Will set focus to the submit button.

To return to a particular row of a multivalue grid use the string '~|' followed by the row number as shown in the following example:

DBRETURN.TO.FIELD = "DBC.CLIENT.CODE~|":DBMVCOUNT-1

When focus is moved from a multivalue field to another multivalue field, using DBRETURN.TO.FIELD, then the '~|' suffix must be used to ensure that focus can be assigned. DesignBais will heed the value in DBMVCOUNT and attempt to move focus to the row designated by the value in DBMVCOUNT. If the source field has more values (rows) than the target field then focus cannot be assigned to any of the rows greater than the maximum row value in the target field.

Therefore if the developer cannot be sure that the number of rows in source and target will be the same, or if moving from a row greater than 1, then use the form DBRETURN.TO.FIELD = 'mvfieldname~|1'. This ensures that if the target field is null then focus will be on the first row.

Moving from the third row, say, of a multivalue field (DBMVCOUNT = 3) to a null multivalue field will only succeed if DBRETURN.TO.FIELD is set to 'mvfieldname~|1'.

To return to a particular cell of an On-Form Report use the format: DBRETURN.TO.FIELD = 'reportName~|row.col'. Report names start with "R." which is the trigger for the OFR cell focus.

Setting DBRETURN.TO.FIELD = "DBNOFOCUS" allows the developer to signal that, following this server hit, focus is to be determined by the browser.

Note that if no value is assigned to DBRETURN.TO.FIELD then focus is determined by the *Suppress Focus* field in Global and System Parameters. Refer to Global Parameters.

To return to an input cell of an On-form report called R.REPORT.EXAMPLE use the following:

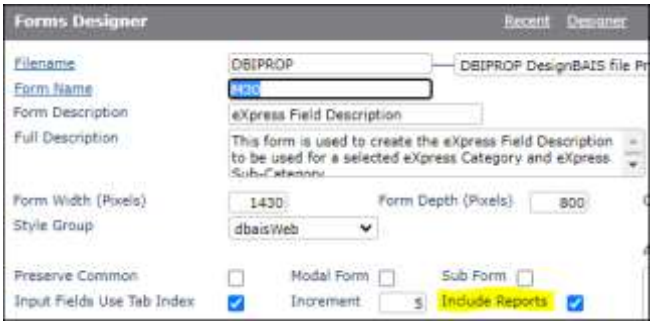
```
DBREPORT.UPDATE = ""
DBREPORT.UPDATE<1,1> = " R.REPORT.EXAMPLE "
DBREPORT.UPDATE<2,1> = ROW
DBREPORT.UPDATE<3,1> = CELL
DBREPORT.UPDATE<6,1> = "I"
DBREPORT.UPDATE<7,1> = 1      ;* this allows the value in the cell to be set to null
* Place cursor back in cell
DBRETURN.TO.FIELD = " R.REPORT.EXAMPLE " : "~|" : ROW : "." : CELL
*  if the cell is a date field and uses Ctrl+Enter then add ".ID" after the cell
DBRETURN.TO.FIELD = " R.REPORT.EXAMPLE " : "~|" : ROW : "." : CELL:".ID"
```

When clicking on an image or a button in a DesignBais form, in the absence of a specific direction to set focus via DBRETURN.TO.FIELD, then focus will return to the button or image that was clicked.

DBRETURN.TO.FIELD should not be used to return to a duplicate field on a form. In most cases the DesignBais auto focus mechanism will pick the correct next field. Use of DBORIG.EVENTSOURCE is also possible.

Set the option *Include Reports* in Forms Designer in order to allow a user to tab through input fields in an OFR.

See Using Dialog Box for IERR.TEXT and DBDS from an On-form Report



DBKEYCODES

This common variable contains the value of the keys pressed by the user. It consists of at least five pipe-separated parts.

FIELD(DBKEYCODES,"|",1) = ascii value of the key
FIELD(DBKEYCODES,"|",2) = ascii value of SHIFT + key
FIELD(DBKEYCODES,"|",3) = "true" if the CTRL key has been simultaneously pressed otherwise “false”
FIELD(DBKEYCODES,"|",4) = ascii value of ALT + key
FIELD(DBKEYCODES,"|",5) = form element id
FIELD(DBKEYCODES,"|",6) = field property name

Use the following to verify if the user has pressed Ctrl+Enter:

CONTROL.ENTER.PRESSED = ((FIELD(DBKEYCODES,"|",3) = "true") + (FIELD(DBKEYCODES,"|",1) = 13) = 2)

The code snip below illustrates how to display the calendar in an on-form report date input field.

```
*
REPORT:
* Process to perform Report cell process
* EVENTSOURCE = Report Name
      ROW = FIELD(DBREPORT.CELL,'.',1)
      COL = FIELD(DBREPORT.CELL,'.',2)
      CONTROL.ENTER.PRESSED = ((FIELD(DBKEYCODES,"|",3) = "true") + (FIELD(DBKEYCODES,"|",1) = 13) = 2)
      BEGIN CASE
         CASE EVENTSOURCE = "R.REPORT3" AND ROW = 0
            NAME = EVENTSOURCE
            SORTJUST = 'L'
            CALL DBI.G.SORT.ONFORMREPORT(NAME,COL,SORTJUST)
         CASE SCREEN.NO="OFRDEMO" AND EVENTSOURCE = "R.REPORT1"
            IF CONTROL.ENTER.PRESSED THEN
               IF COL=2 THEN
                  DBRETURN.TO.FIELD = "R.REPORT1~|":ROW:".2.ID"
                  PROCESS.STACK="DBIPARMS_CALENDAR"
               END
            END
      END CASE
```

Using the timer function

A DesignBais form can be periodically refreshed if required by the user of the DBTIMER variable.

DBTIMER

DesignBais will refresh the current form by the duration indicated if DBTIMER is set to a non-zero value. This value is in milliseconds and should not be set to a value less than 15 seconds. Problems may occur if the timer duration is less that the time it takes to refresh the form.

When the form is refreshed by the timer function, the “After Display” event for the form will be triggered. There is no “timer event” so your application code needs to differentiate between the *After Display* event which is fired after the initial form display and the *After Display* event fired on a DBTIMER timeout. See note below.

Usage: DBTIMER = *Millisecond setting*
Eg. DBTIMER = 15000 ; * 15 seconds

DBTIMER supports an additional setting that will invoke the database server, but not force a refresh or an AFTER DISPLAY event.

The event is TIMER.

Usage:

DBTIMER<1,1> = Timer repeat time in milliseconds
DBTIMER<1,3> = “TIMER”
DBTIMER<1,4> = Program name to call when timer is activated
DBTIMER<1,5> = Eventsource to set when program is called
DBTIMER<1,6> = Description to display on progress bar
DBTIMER<1,7> = Percentage complete
DBTIMER<1,8> = Value entered here will be passed back in PROCESS.PARAMETER

Eg.

DBTIMER<1,1> = 15000
DBTIMER<1,3> = “TIMER”
DBTIMER<1,4> = DB.I.DBCLIENT
DBTIMER<1,5> = “MyTimer”
DBTIMER<1,6> = “Loading…”
DBTIMER<1,7> = 25
DBTIMER<1,8> = “MY.PARAM”

In the above example, the timer event will be invoked every 15 seconds, the program DB.I.DBCLIENT will be called. The PROCESS.EVENT is “TIMER” and the PROCESS.EVENTSOURCE is “MyTimer”.

Note for developers. If DBTIMER is used to keep track of a phantom process then the *After Display* event may be in use to check if the phantom is already running as well as initialising phantom progress controls, and initialising form section collapse controls. A scenario may arise where the phantom completes and the results are displayed by, for example, by revealing a collapsed form section containing an On-form Report. If the final *After Display* event triggered by DBTIMER resets the form section collapse controls then you may see the On-form Report appear briefly then disappear based on

a section collapse. You will need to add some logic in the *After Display* code to differentiate between the initial event and that triggered by the completion of the timer event.

The following code snippets will assist developers to implement a progress bar using DBTIMER.
In the first example the progress bar displays in the DBWORK field. This is because DBTIMER<1,6> and <1,7> are left null.

```
      IF NUM(OS.DATAOUT) THEN
         * PHANTOM has started
         DBTIMER = 5000
         DBTIMER<1,3> = "TIMER"
         DBTIMER<1,4> = "DBI.I.RV"
         DBTIMER<1,5> = "PHANTOM_TRACK_":PH.ID
         PH.PERC = 0
         PH.MSG = 'Phantom Started'
         DBWORK<DBCK.CF.PROGRESS.WK> = '<progress class="dbaisSearchLabelBlue" value="':PH.PERC:'"
style="width:200px;"></progress>'
         DBWORK<DBCK.CF.PROGRESS.MSG.WK> = PH.MSG
      END
```

The presence of a value in DBTIMER<1,6> triggers the display of the different progress bar that appears on top of the application form. The value in DBTIMER<1,7> is calculated by the developer and represents the % progress of the function. We suggest that the value commences at 5 rather than 0.

```
      IF NUM(OS.DATAOUT) THEN
         * PHANTOM started
         DBTIMER = DBTIMER.INCREMENT
         DBTIMER<1,3> = "TIMER"
         DBTIMER<1,4> = "DBI.I.UPGRADE"
         DBTIMER<1,5> = "PHANTOM_TRACK_":PH.ID
         DBTIMER<1,6> = 'Processing...'
         DBTIMER<1,7> = 5
         PH.PERC = 0
         PH.MSG = 'Phantom Started'
         DBWORK<DBIPM.U.PROGRESS.MSG.WK> = PH.MSG
      END
```

The code below is an example of the updating of the value that indicates the progress of the function.

```
*
UPDATE.PHANTOM.PROGRESS:
*
      IF DBWORK<DBIPM.U.SELCNT.WK>+0 = 0 THEN
         PH.PERC = 100
      END ELSE
         PH.PERC = INT(DBWORK<DBIPM.U.CNT.WK>*100/DBWORK<DBIPM.U.SELCNT.WK>)
      END
      DBTIMER<1,7> = PH.PERC

      READ PH.REC FROM F.DBISTATS,PH.ID ELSE PH.REC = ''
      PH.REC<DBISA.PH.PERCENTAGE> = PH.PERC
      PH.REC<DBISA.PH.MESSAGE> = UPGRADE.OPTIONS<OPTNO>
      PH.REC<DBISA.PH.SELECTED>= DBWORK<DBIPM.U.SELCNT.WK>
      PH.REC<DBISA.PH.PROCESSED>= DBWORK<DBIPM.U.CNT.WK>
      PH.REC<DBISA.PH.STATUS>= 'A'
      WRITE PH.REC ON F.DBISTATS,PH.ID
      RETURN
```

Using DBTIMER to verify if a file exists

DBTIMER can be used to check if a relative path (file) exists. The DBDEMO_TIMER form in the DesignBais *Demo Forms* list demonstrates this function.

There are two new events *TIMEOUT* and *FILEEXIST*:

```
CASE THIS.EVENT = "TIMEOUT"
   GOSUB TIMER.PARA
CASE THIS.EVENT = "FILEEXIST"
   GOSUB FILEEXIST
```

The traditional TIMER funtion runs continuously in the browser until it is stopped. In contrast the *TIMEOUT* event will run once in the browser. The developer can trigger it again until the DBTIMER variable is cleared in the *FILEEXIST* event. Or the developer can set DBTIMER<1,3> = "TIMEOUTEND" to kill the current *TIMEOUT* event and this also clears DBTIMER.

To trigger the TIMEOUT event set DBTIMER:

```
DBTIMER = “”
DBTIMER<1,3> = “TIMEOUT”
```

The FILEEXIST event checks if a file exists on the web server. Place the following code in a button event, for example, in order to check if a file (the file / pathname is in DBWORK<DEM.FILE.PATH.WK> in this example):

```
DBTIMER = “”
DBTIMER<1,3> = 'FILEEXIST'
DBTIMER<1,4> = 'DBI.I.DEMO'                    the name of the process (subroutine) to process this event
DBTIMER<1,5> = 'CHECK.FILE'                    the EVENTSOURCE in the FILEEXIST event
DBTIMER<1,8> = DBWORK<DEM.FILE.PATH.WK>        contains the file path to be checked
```

The result of this will be returned to the specified subroutine process with PROCESS.EVENT set to *FILEEXIST*. The result of either *True* or *False* is passed in DBVALUE and the name of the file path that has been checked (the value passed in DBTIMER<1,8>) is returned in PROCESS.PARAMETER.

```
FILEEXIST:
  BEGIN CASE
     CASE SCREEN.NO = "TIMER" AND EVENTSOURCE = 'CHECK.FILE'
        DBWORK<DEM.FILE.EXISTS.WK> = DBVALUE
        VERIFY.FILE = THIS.PARAMETER
        DBTIMER = ''
  END CASE
  RETURN
```

To end the TIMEOUT process set DBTIMER:

```
DBTIMER = “”
DBTIMER<1,3> = 'TIMEOUTEND'                    this will kill currently running timeout
```

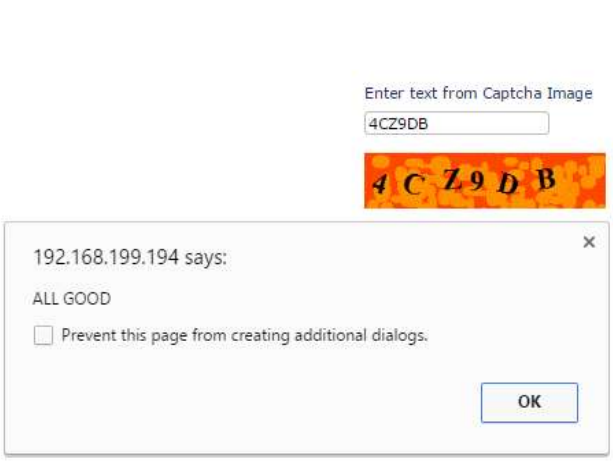
Using DBTIMER to check if a file has been created in order to allow the user to proceed:

- Display a “Please wait” message
- Start the new TIMEOUT function
- In the TIMEOUT event use FILEEXIST to prompt DesignBais to check if the file exists
- In the FILEEXIST event processing check DBVALUE
- If DBVALUE = “False” then set another TIMEOUT
- If DBVALUE = “True” then clear DBTIMER
- There could be a button to kill the TIMEOUT if this is considered necessary

Using the Captcha function

DBCAPTCHA

This is an example of how one might implement the Captcha feature:



Set up an input field and a Captcha field. Place a Process After call to your subroutine on the input field.

After the characters displayed in the captcha image are entered in the input field the validation event will fire.

The Process After event will be called twice.

Intially it will be called with PROCESS.PARAMETER set to null.
The second time DesignBais will set PROCESS.PARAMETER = “GETCAPTCHA”.

So as shown in the sample code below on the second time the Process After validation event in your subroutine must set DBCAPTCHA = DBVALUE to capture the details for DBCAPTCHA.STATUS.

```
CASE SCREEN.NO = "CAPTCHA" AND EVENTSOURCE = "DEM.CAPTCHA.TEXT.WK"
   IF PROCESS.PARAMETER # "GETCAPTCHA" THEN
      DBCAPTCHA = DBVALUE
      RETURN
   END
   * DBVALUE contains the string entered in "DEM.CAPTCHA.TEXT.WK" for validation
   * DBCAPTCHA.STATUS contains details of the CAPTCHA field and the results - see the GOSUB
   *
   GOSUB CAPTCHA.DISPLAY
```

DBCAPTCHA.STATUS is returned with 4 values:
<1,1> The name of the subroutine in the Process After slot.
<1,2> The name of the input field on the form.
<1,3> 0 (zero)
<1,4> 0 (zero) or 1 [1: the input string did not match the image; 0: input matched the image]

Your subroutine can use DBCAPTCHA.STATUS<1,4> to determine if a match was found and process accordingly. In the example above the program contained the line:

IF DBCAPTCHA.STATUS<1,4> = “0” THEN DBDS<-1> = “ALL GOOD”

Tracking events and event processing

DesignBais provides a number of common variables that allow the developer to track what event occurred on the client and when.

PROCESS.EVENT

The PROCESS.EVENT variable determines what condition occurred on the Client form. These events are triggered every time something occurs on the client that requires the server to be involved.

PROCESS.EVENT is a read-only variable and should not be modified programmatically

Please see below for a complete set of values for PROCESS.EVENT

PROCESS.EVENTSOURCE

PROCESS.EVENTSOURCE is used to determine what field or form was responsible for the event being triggered.

The values associated with PROCESS.EVENTSOURCE vary depending on which type of event is being fired.

Please see below for an explanation of the PROCESS.EVENTSOURCE for each PROCESS.EVENT

PROCESS.PARAMETER

PROCESS.PARAMETER is used to provide extra detail for the event. In most cases PROCESS.PARAMETER is defined at design time, though there are a number of events that DesignBais assigns a value to PROCESS.PARAMETER.

Please see below for an explanation of the PROCESS.EVENTSOURCE for each PROCESS.EVENT

Events that are tracked through PROCESS.EVENT

"AFTER DELETE" From the **Process After Write** field in the Updating section of the form designer.

Only set when the *Delete Record* update type is being used.

PROCESS.EVENTSOURCE = Field Name that invoked the deletion (usually a button)
PROCESS.PARAMETER = Assigned in forms designer

"AFTER DISPLAY" From the **Process After Display** field on main the forms designer form.

This event occurs after the form has been displayed, but before data is displayed. It is also fired to signal a DBTIMER event.

PROCESS.EVENTSOURCE = Not Applicable
PROCESS.PARAMETER = Assigned in forms designer

"AFTER READ" From the **Process After Read** field in the properties of a field. This event occurs after a selection is processed. This provides the ability to further refine the selection list before presenting the results to an end-user.

This event occurs after a record has been read.

PROCESS.EVENTSOURCE = Input Field Name that invoked the read
PROCESS.PARAMETER = Assigned in forms designer

"AFTER SELECT" From the **Process After Selection** field in the selection process.

This event occurs after the user has made a selection from the selection form. The selected value is available in DBVALUE. Multiple-selections are returned as multiple values in DBVALUE.

PROCESS.EVENTSOURCE = Form Name of the Selection (*Filename_Formname)*
PROCESS.PARAMETER = Assigned in forms designer

"AFTER WRITE" From the **Process After Write** field in the Updating section of the form

Only set when the *Write Record* update type is being used.

PROCESS.EVENTSOURCE	= Field Name that invoked the write (usually a button)
PROCESS.PARAMETER	= Assigned in forms designer

"BEFORE CLOSE" From the "**Subroutine on Main form Close**" field on the Menu Definition

If a menu structure is invoked that has a subroutine named in the "Subroutine on Main form Close" slot, any main form close event will invoke the named subroutine.

The subroutine named can be used to perform any housekeeping that your application requires when the main form is closed. Sub menus will inherent the subroutine named in parent menus.

PROCESS.EVENTSOURCE	= SCREENROOT

"BEFORE DELETE" From the **Process Before Write** field in the Updating section of the form designer.

Only set when the *Delete Record* update type is being used.

PROCESS.EVENTSOURCE	= Field Name that invoked the deletion (usually a button)
PROCESS.PARAMETER	= Assigned in forms designer

"BEFORE FIELD" From the **Process Before** field in the properties of a field. This event is invoked when an input field gains focus.

PROCESS.EVENTSOURCE	= Input Field Name that invoked the event
PROCESS.PARAMETER	= Assigned in forms designer

"BEFORE DISPLAY" From the **Process Before Display** field on main the forms designer form.

This event occurs before the form has been displayed.

PROCESS.EVENTSOURCE	= Not Applicable
PROCESS.PARAMETER	= Assigned in forms designer

This event is designed to be used if there is a requirement to switch SCREENROOT before the form loads. This should be performed using: PROCESS.STACK = *New form*

"BEFORE READ" From the **Process Before Read** field in the properties of a field. This event occurs before a record read takes place. This event can be used to change values of key fields before the actual read is processed.

PROCESS.EVENTSOURCE	= Input Field Name that invoked the read
PROCESS.PARAMETER	= Assigned in forms designer

"BEFORE REPORT" This event is triggered before the execution of a DesignBais Report
It will only be triggered if the 'Process Before Report' slot is filled in the Report Designer Main form.

PROCESS.EVENTSOURCE	= Name of the report
PROCESS.PARAMETER	= Not assigned
DBUSAGEVAR	= Contains the values entered in the report run form.

The values in DBUSAGEVAR are offset by 10 fields. The 11th field contains the results of the first input field on the run form, the 12th, the second and so on.

Setting IERR.TEXT to a value other than null in the event handler cancels the report.

"BEFORE SCREEN " From the "**Subroutine to invoke before screen**" field on the Menu Definition

If a menu structure is invoked that has a subroutine named in the “Subroutine to invoke before screen” slot, all screens invoked from that point on will invoke the named subroutine.

PROCESS.EVENTSOURCE is SCREENROOT (The form being loaded from the menu)

“BEFORE SELECTION” From the **Process Before Selection** field in the selection process. This event can be used to validate user input on a selection form before the selection statement is processed.

PROCESS.EVENTSOURCE	= Form name of the Report or Selection Process
PROCESS.PARAMETER	= Form name of the Selection Process

“BEFORE WRITE” From the **Process Before Write** field in the Updating section of the form

Only set when the *Write Record* update type is being used.

PROCESS.EVENTSOURCE	= Field Name that invoked the deletion (usually a button)
PROCESS.PARAMETER	= Assigned in forms designer

“BUTTON” This event is triggered when a button is clicked

PROCESS.EVENTSOURCE	= Button Name
PROCESS.PARAMETER	= Assigned in forms designer

"DBFINDEX KEY" The name of the program can be added to this field.

PROCESS.EVENT	= “DBFINDEX KEY”
PROCESS.EVENTSOURCE	= Name of the file
PROCESS.PARAMETER<1>	= Record Identifier
DBVALUE	= Value of the field

If the program sets the variable IERR.TEXT to anything other than null, the record to be indexed will be skipped.

The value returned in DBVALUE will be the value of the key.

"DBFINDEX LIST" The index definition may not be for a valid file or key structure. You may need a program to derive the keys. This can be useful when you are returning list items for standard lookups. It can be useful in identifying forms or application entries that the user has access to.

"DBFINDEX DISPLAY" The display of an indexed field may be modified before it is displayed in the routine DBI.G.DISPLAY.DBFINDEX. If so add the name of the routine in this field.

PROCESS.EVENT	= "DBFINDEX DISPLAY"
PROCESS.EVENTSOURCE	= Name of the index being displayed
PROCESS.PARAMETER	= Name of the field being displayed
DBVALUE	= Value of the display. You may modify DBVALUE.
DBKEY	= The id of the indexed item

"DBFINDEX STRING" You may wish to call a subroutine to modify the default strings that are passed to the menu build routine.

PROCESS.EVENT	= "DBFINDEX STRING"
PROCESS.EVENTSOURCE	= Index Name
DBVALUE<1>	= String to Send

DBVALUE<2> = Process Description
DBVALUE<3> = Pass Selected to Field

You may modify the values in DBVALUE.

If you set IERR.TEXT (to a non-null value), the click event will be ignored.

"DERIVED"

This event is triggered when a parent field to a derived field changes.

PROCESS.EVENTSOURCE = Field Name to be derived
PROCESS.PARAMETER = Assigned in forms designer
DBMVCOUNT
If the field is a multivalue, DBMVCOUNT contains the value of the current multivalue row. When a derived routine is called as a result of a change to a parent input field, DBVALUE will have been updated into the appropriate DBRECORD, DBOTHER.RECORD(*n*) or DBWORK variable.

"DERIVED KEY"

This event occurs when a derived key is to be determined programmatically. From the **After Display** field on main the forms designer form.

This event occurs after the form has been displayed, but before data is displayed.

PROCESS.EVENTSOURCE = Not Applicable
PROCESS.PARAMETER = Assigned in forms designer

This provides the capability for the developer to change the value of a Derived Key on a form before the derived record is read and the form displayed. Enter V: in the *Default Key Value* field in Forms Designer. In the subroutine defined in the *Process After Display* create the *DERIVED KEY* event and assign the required key value to *DBVALUE*.

"DIALOG"

This event occurs when a user responds to a dialog box question. Please see the section **Creating a dialog box** for a full explanation of dialog box set-up.

PROCESS.EVENTSOURCE = Event parameter assigned to the Dialog
PROCESS.PARAMETER = Value of the Dialog button pressed

"GET SCREEN"

When a request to load a new form is received by the DesignBais server this event will be raised if a subroutine name has been entered into User Group or User record against the login account. This provides the ability to programmatically change requested screens, or even restrict access to screens at certain times.

PROCESS.EVENTSOURCE = The username that invoked the screen
PROCESS.PARAMETER = The user group of the user
SCREENROOT = The requested form

The requested screen name may be changed in the called subroutine by setting PROCESS.STACK to a *Filename_Formname* combination.

"HTMLEDIT"

This event is raised when the HTML editor is closed. See HTMLEDIT in common variable usage.

PROCESS.EVENT = "HTMLEDIT"
PROCESS.EVENTSOURCE = DBRTFRESPONSE<1,2>

"IMAGE"

This event is raised when an image is clicked on a form at run time. This event will only be raised in the image has a 'Process After' defined.

PROCESS.EVENTSOURCE = Name of the image that invoked the event.
PROCESS.PARAMETER = Assigned in forms designer

This provides the ability for the developer to trap image click events.

"MOUSE OVER" This event is raised for an image when a mouseover occurs. Please see section control for more details.

PROCESS.EVENT	= "MOUSE OVER"
PROVESS.EVENTSOURCE	= *ImageName.ext*

"MODAL RETURN" From the **Modal Form Return Process** on the form designer. This event occurs when a modal form is closed as control is returned to the base form. The process to call must be defined on the base form not the modal form being closed.

PROCESS.EVENTSOURCE	= Name of the form that closed (*Filename_Formname)*
PROCESS.PARAMETER	= Not Used

This provides the ability for the developer to trap close events when returning to the main form.

"MODAL RETURN HEADER"
This event will trigger if the header form has the Modal Form Return Process specified. This will be invoked before the "MODAL RETURN" in the current form.

"MV_ADD" This event is triggered when a multi-value row is added within a multi-value control set. This event will only be triggered if there is an Add Process in the Multivalues sub-form within the form designer.

PROCESS.EVENTSOURCE	= Name of the Multivalue control set
PROCESS.PARAMETER	= Assigned in forms designer
DBMVCOUNT	= Last row in the current array

"MV_DELETE" This event is triggered when a multi-value row is deleted within a multi-value control set. This event will only be triggered if there is a Process after delete in the Multivalues sub-form within the form designer.

PROCESS.EVENTSOURCE	= Name of the Multivalue control set
PROCESS.PARAMETER	= Assigned in forms designer
DBMVCOUNT	= Row to be deleted

"MV_HEADER" This event is triggered when a multi-value column header is clicked. This event is only triggered if the Header Process is assigned within the Multi-value field properties within the forms designer.

PROCESS.EVENTSOURCE	= Name of the Multivalue field
PROCESS.PARAMETER	= Assigned in forms designer
DBMVCOUNT	= Last active row (row that currently has focus). If no row has focus then the value is 1

"MV_INSERT" This event is triggered when a multi-value row is inserted within a multi-value control set. This event will only be triggered if there is a Process after insert in the Multivalues sub-form within the form designer.

PROCESS.EVENTSOURCE	= Name of the Multivalue control set
PROCESS.PARAMETER	= Assigned in forms designer
DBMVCOUNT	= Row number of the insertion point.

"REPORT" This event is triggered when an active cell on an On-form Report is clicked. This event is only triggered if the On-form Report has a Process After assigned.

PROCESS.EVENTSOURCE	= Name of the On-form Report
PROCESS.PARAMETER	= Assigned in forms designer
DBVALUE	= Contains the value assigned to OUTPUT.KEYS in the clicked cell
DBREPORT.CELL	= *Row.Col*

The row and column of the clicked cell is separated by a period ('.').

"REPORT BREAK" This event is triggered when a break occurs during a DesignBais report execution.

This event will only be triggered on a break if a subroutine name has been defined in the 'Subroutine to call when the break is activated' slot is filled in the report designer main form.

PROCESS.EVENTSOURCE	= Name of the break
PROCESS.PARAMETER	= List of accumulators names (value mark delimited)
DBVALUE	= List of accumulators (value mark delimited)

Note that DBBREAKTOTALS is only populated if the report has 'Two Pass Report' checked.

"REPORT READ" This event is triggered after every read in a DesignBais report. This pertains only to reports built using the DesignBais report designer. It is not triggered in an On-form Report.

This event will only be triggered if the 'Process After Read' slot is filled in the report designer main form.

PROCESS.EVENTSOURCE	= Name of the report
PROCESS.PARAMETER	= Not assigned
DBVALUE	= The identifier of the read record.
DBRECORD	= The contents of the record read.

"VALIDATE" This event is triggered when the value of a cell changes on a DesignBais form. This event will only be triggered if the 'Process After' slot is filled on the properties for a field in the forms designer.

PROCESS.EVENTSOURCE	= Input Field Name that invoked the event
PROCESS.PARAMETER	= Assigned in forms designer
DBVALUE	= Value of the entry
DBMVCOUNT	= If the field is a multivalue DBMVCOUNT contains the value of the current multivalue row.

If IERR.TEXT is set to anything other than null, DesignBais will display an error dialog at the completion of the named subroutine. The value entered into the cell that triggered the event will not be added to a record.

The process name can be either a cataloged subroutine or a form id (*Filename_Formname* optionally followed by *~E ~L ~S or ~M* or a combination of these). In a Multivalue Grid where you wish to call another form from the *Process After* slot it may be necessary to call a subroutine and within the subroutine code set *PROCESS.STACK = "Filename_Formname"*, rather than place the form id directly in the slot.

"UPLOAD" This event is triggered as a return from the file upload process. It Is triggered when the upload is complete.

Please see the Chapter *File Upload Process* later in this document.

"EXPRESS LINK" This event is used to calculate the sub-file links in DesignBais eXpress. Please see the Chapter titled, DesignBais eXpress later in this document

"BEFORE READWRITE" This event is used if a global read/write event has been nominated. DBIGLOBAL file must be created.

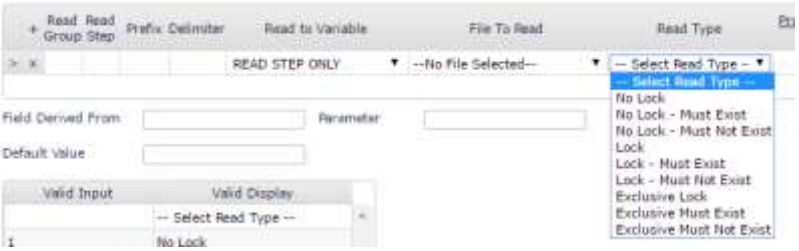
File DBIGLOBAL
Item READWRITE
Attr 1 Contains the name of the subroutine to be called.

All DesignBais reads or writes will call the nominated subroutine.

The subroutine will be called with the following parameters

PROCESS.EVENT = “BEFORE READWRITE”
PROCESS.EVENTSOURCE = Name of the file being accessed
PROCESS.PARAMETER = Record id : VM : Process Type

Process Type corresponds to the Read Type dropdown list in Forms Designer Read Group set up for read and update processes, and has a value of 6 for delete. See below.



+	Valid Input List	Description
> x		-- Select Read Type --
> x	1	No Lock
> x	2	No Lock - Must Exist
> x	4	No Lock - Must Not Exist
> x	3	Lock
> x	8	Lock - Must Exist
> x	9	Lock - Must Not Exist
> x	10	Exclusive Lock
> x	11	Exclusive Must Exist
> x	12	Exclusive Must Not Exist

```
value>-- Select Read Type --</option>
value="1">No Lock</option>
value="2">No Lock - Must Exist</option>
value="4">No Lock - Must Not Exist</option>
value="3">Lock</option>
value="8">Lock - Must Exist</option>
value="9">Lock - Must Not Exist</option>
value="10">Exclusive Lock</option>
value="11">Exclusive Must Exist</option>
value="12">Exclusive Must Not Exist</option>
```

In addition there are Form read type options on the Forms Designer front screen:

```
value="1">No Lock</option>
value="2">No Lock - Must Exist</option>
value="3">Lock</option>
value="10">Exclusive Lock</option>
value="11">Exclusive Lock Must Exist</option>
```

and Update Type options on the Forms Designer ‘Updating’ button:

```
value="4">Write / Release</option>
value="5">Write / No Release</option>
value="6">Delete Record</option>
```

“AFTER READWRITE”

This event is used if a global read/write event has been nominated. DBIGLOBAL file must be created.

File DBIGLOBAL
Item READWRITE
Attr 1 Contains the name of the subroutine to be called.

All DesignBais reads or writes will call the nominated subroutine after completion.
The subroutine will be called with the following parameters

PROCESS.EVENT = “AFTER READWRITE”
PROCESS.EVENTSOURCE = Name of the file being accessed
PROCESS.PARAMETER = Record id : VM : Process Type

See **Process Type** above for a list of Process Type codes

"KEYPRESS" If the field has been setup to invoke the server on a Keypress Event, DesignBais will contact the database server after 300ms of keyboard inactivity. The field must have a process after subroutine defined in order for the a change in the field value to be retained. The sequence generated is a keydown event, a blur event and a button click. During the keydown the amended field value is displayed in DBVALUE but is not sent to the read variable, such as DBRECORD. During the blur the original value is re-instated.

PROCESS.EVENT = "KEYPRESS"
PROCESS.EVENTSOURCE = The Input Field
DBVALUE = The value of the input at the time of the call

"REMOTERETURN" This event is triggered on return from a call to web service.
The program to call and the eventsource are defined in attributes 12 and 13 of DBCALLWS.

Example:

Calling the web service set these attributes
DBCALLWS<12> = "DB.I.DBCLIENT" Program to invoke on return
DBCALLWS<13> = "B.LOADRATE" Eventsource

On return from the web service these common variables allow the result to be processed
PROCESS.EVENT = "REMOTERETURN"
PROCESS.EVENTSOURCE = "B.LOADRATE"
PROCESS.PARAMETER = contains the output from the web service

"DERIVED KEY" This event is triggered following the AFTER READ event when a form has a Default Key Value.

"REINSTATE.SESSION" This event is triggered on return from an external wesite. Refer to Session Reinstatement.

"REINSTATE.SESSION.POST"
This event is triggered on return from an external wesite. Refer to Session Reinstatement.

Use the REINSTATE.SESSION.POST to perform updates to DBRECORD etc.

"PHANTOM" This event is triggered by the DBI.P.CALLDBSUB subroutine:
CALL DBI.P.CALLDBSUB Sub.Name EventSource EventParameter

PROCESS.EVENT = "PHANTOM"
PROCESS.EVENTSOURCE = EventSource
PROCESS.PARAMETER = EventParameter

CALL @Sub.Name

"LOOKUP" Refer to the DesignBais Responsive Design Manual.

"LOOKUP LABEL" Refer to the DesignBais Responsive Design Manual.

A value can be passed to PROCESS.PARAMETER via the PROCESS.STACK command. This is achieved by appending a third tilde ('~') to the value passed via PROCESS.STACK, followed by the value to which PROCESS.PARAMETER is to be set.

Eg. PROCESS.STACK = "DBCLIENT_LESSON1~L~E~NEXT"

The form DBCLIENT_LESSON1 will be invoked as a modal, layered form in inquiry mode and PROCESS.PARAMETER will be set to a value of "NEXT".

Switching Files at Runtime (Selections, Forms and Reports)

DesignBais provides a number of common variables that allow the design file to be switched at runtime.

In the “BEFORE SELECTION” event, the developer may wish to switch the name of the file being used for the selection with another filename.

This is achieved by using the common variables:

DBSWITCHSELECTFILE

The File to change from

DBSWITCHSELECTFILETO

The file to change to

Usage:
DBSWITCHSELECTFILE = *Switch from DBCLIENT file*
DBSWITCHSELECTFILETO = *Switch to file*
Eg.
DBSWITCHSELECTFILE = “DBCLIENT” ; * Switch from DBCLIENT file
DBSWITCHSELECTFILETO = “DBCLIENT.ARCHIVE” ; * Switch to file

There is another common variable that can be used in conjunction with the previous two variables, named DBSELECTAPPEND. This provides the ability to add the following types of statements to the selection process at runtime.

DBSELECTAPPEND

Usage
DBSELECTAPPEND = *Append to Sentence*
Eg.
DBSELECTAPPEND = “ USING DICT DBCLIENT”

In the above example the Selection process would select items from DBCLIENT.ARCHIVE and append “USING DICT DBCLIENT” to the selection statement. Remember to clear DBSELECTAPPEND if you are processing multiple file selections and, for example, the ‘USING DICT’ clause has to change or be removed.

You can also change any DesignBais internal reads on a form or a report by using the following COMMON variables

DBSWITCHGROUP

The read group to change

DBSWITCHFILE

The new file name

Typically this would be used in the “BEFORE READ” event

Usage:
DBSWITCHGROUP = *ReadGroup*:{Value Mark Delimiter}:*ReadGroup*
DBSWITCHFILE = *Filename*:{Value Mark Delimiter}:*Filename*
Eg.
DBSWITCHGROUP = 2:VM:4 – Read Groups 2 and 4 will change
DBSWITCHFILE = “DBCLIENT.ARCHIVE”:VM:”DBCLIENT.SALES.ARCHIVE”

This could also be used to read sub-files where the open is FILENAME,SUBFILENAME

Changing an Image on a form or report at runtime

DesignBais provides the ability to change multiple images that are being displayed on a form. This would usually be performed in an 'After read' event or a 'Validate' event

DBIMAGESPEC

In order to change an image or images on a form, you must first identify which image you want to change. This is noted in the first attribute of DBIMAGESPEC

Usage: DBIMAGESPEC<1> = *"From Image"*
DBIMAGESPEC<2> = *"To Image"*

Eg. DBIMAGESPEC<1> = "DBLOGO.JPG"

The second attribute of DBIMAGESPEC is reserved for the new name for the image.

Eg. DBIMAGESPEC<2> = "NEWIMAGE.JPG"

In the above to examples DesignBais will replace the image DBLOGO.JPG with NEWIMAGE.JPG

Eg. In an after read event you may want to change the image to an image name in the record.

DBIMAGESPEC<1> = "DBLOGO.JPG" ; * Name of the image to change
DBIMAGESPEC<2> = DBRECORD<DBC.IMAGE.NAME> ; * Name associated with the new image

DBIMAGESPEC can be multivalued if you wish to change more than one image at a time.

Eg. DBIMAGESPEC<1> = "DBLOGO.JPG":VM:"DBLOGO2.JPG"
DBIMAGESPEC<2> = "NEWIMAGE.JPG":VM:"NEWIMAGE2.JPG"

Always refer to the image name that was originally added to the form in forms designer as the image to change.

There are 2 additional attributes.

DBIMAGESPEC<3> = multivalued help text for mouse over image (title property)
DBIMAGESPEC<4> = multivalued text to display if image is missing (alt attribute)

The alt attribute specifies an alternate text for an image, if the image cannot be displayed. If DBIMAGESPEC<4> is null then DBIMAGESPEC<3> is used in the case that the image cannot be displayed.

Calling a URL from a BASIC subroutine

Provides the ability to call a URL from any event.

DBCALLURL

Usage: DBCALLURL = "your_url"
Eg. DBCALLURL="http:\\www.google.com;newWindow"

Default behaviour is to run in the current tab.
Add the ";*newTab*" option to run the url in a new tab in the same browser window.
Add the ";*newWindow*" option to run the url in a new browser window. This opens another instance of the browser.

Omitting *";newTab"* or *";newWindow"* replaces the user's current window with the URL specified. **Note however that this does not close the user's session or release their license. The license will be released automatically after 20 minutes. Exclusive locks and other resources held in the session will not be automatically released for approximately 12 hours. Therefore, the application code should logout the user as part of this process.**

A combination of URL Encoding and HTML Encoding may be required to build a valid URL. Example, an ampersand in a data field must be URL Encoded into "%26" but a query string separating data fields is separated by an ampersand using "&".

It is possible to call an external website which disables your base DesignBais form using the “externalDBModal” flag.

Usage: DBCALLURL = “your.url;externalDBModal”
Eg. DBCALLURL = “example.url?query.string;externalDBModal”

The url and query string are of no consequence to DesignBais. Return from the external website must be via “return.aspx”

When DesignBais regains control it will be via MODAL.RETURN with EVENTSOURCE = “externalModal”.

The options ;*newWindow* and ;*parentWindow* (should be considered case-sensitive for cross-platform compatibility) are not used and are stripped out by DesignBais.

The DesignBais Data Component sends direct javascript commands to open urls. The following scenarios are catered for:

1. Open the url in the same window: *window.location = "http://some_url"*. This is preferred over the similar *window.open("http://some_url", "_self")*.

2. Open the url in a new window (or tab): *window.open("http://some_url", "_blank", specs, replace)*

3. Open the url in parent's window (this assumes that DesignBais is in an IFRAME): *window.open("http://some_url", "_parent", specs, replace)*

4. Open the url in top window (this assumes that DesignBais is in an IFRAME buried deep in the hierarchy of a set of IFRAMEs in a TOP window): *window.open("http://some_url", "_top", specs, replace)*

Notes:
- (2): "_blank" can be omitted because it's default: New window.
- (3) & (4) do the same thing if the DesignBais IFRAME is an immediate child of a window.
- *"specs"* is an optional parameter that specifies window dimensions etc.. Developers can populate this is required.
- *"replace"* is (true or false). It's an optional parameter that specifies whether the URL creates a new entry or replaces the current entry in the history list. Rarely used. Developers can populate this is required.

The DesignBais engine uses *top.location*. This changes the top window location whether or not the application is in an IFRAME. To open new windows DesignBais uses *window.open()*. This opens new windows whether or not the application is in an IFRAME.

The rare scenario when you want to change the location of the immediate parent IFRAME which itself is an IFRAME in a window (i.e. DesignBais is the grandchild) would normally use *"_parent"* or ‘*parent.window.location=””*’ to change the location of the immediate parent.

The query string must adhere to the conventions / rules for query strings. They are defined to be *name=value* pairs separated by *&*. The *&* acts like a AM, VM or SVM in a dynamic array. There must be an = sign within each segment delimited by the *&* character.

Calling a Web Service from a BASIC subroutine

Provides the ability to call a web service.

DBCALLWS

Refer to the chapter Invoking a Web Service where the method is documented.

Changing the Display Attributes at runtime

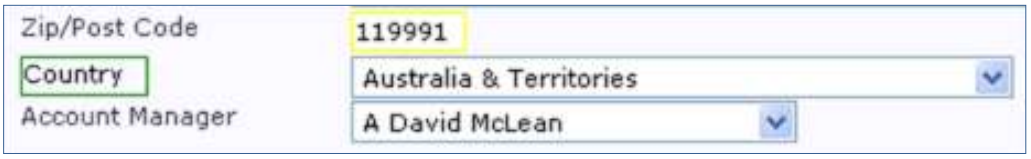
DBSTYLESPEC

This variable provides the developer the ability to change the styles of any field or control placed on a form.

DBSTYLESPEC can be multivalue delimited to affect multiple fields/controls

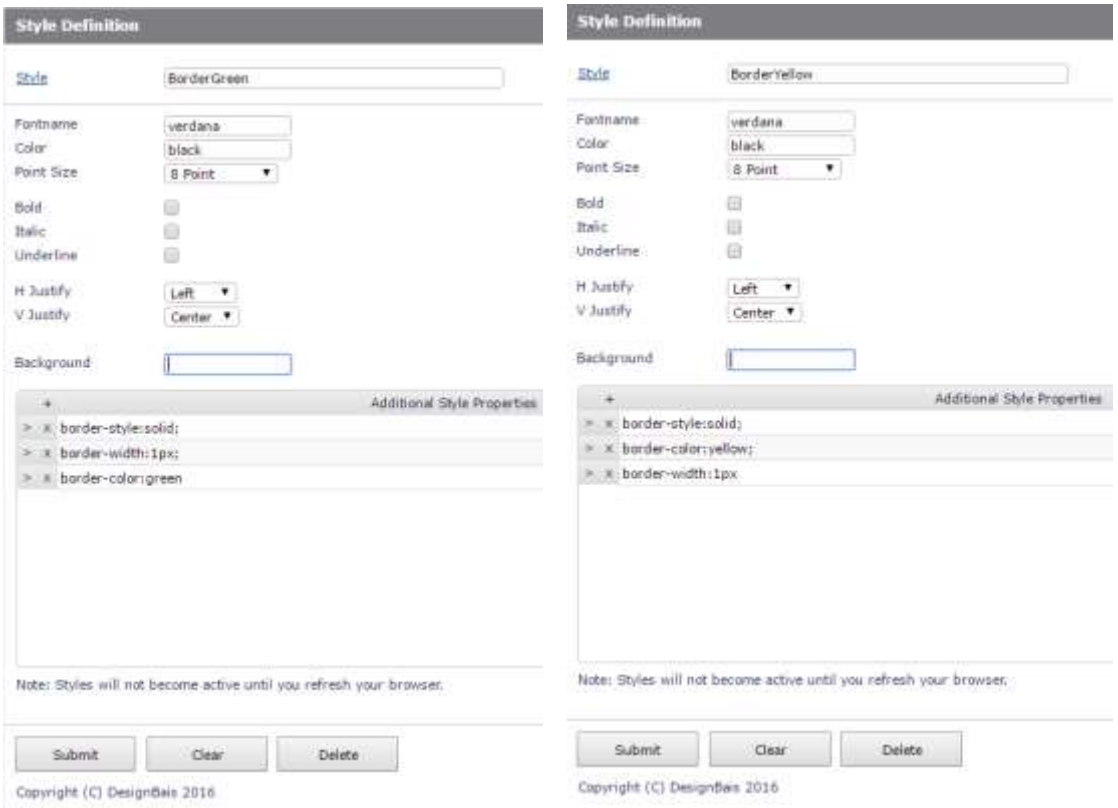
Usage DBSTYLESPEC<1> = Name of Field/Control (MV list)
DBSTYLESPEC<2> = Name of the Style Definition Record (MV list)
DBSTYLESPEC<3> = List of Part Types (MV list)
[B=Border (outside div), R=Row (tr), C=Cell (td), I=Input element]
DBSTYLESPEC<4> = List of Row Numbers (MV list)
DBSTYLESPEC<5> = List of Cell Numbers (MV list)

Eg. DBSTYLESPEC = "Country":VM:"DBC.PCODE"
DBSTYLESPEC<2> = "BorderGreen":VM:"BorderYellow"



Both the BorderGreen and BorderYellow are Style Definition records.

The named entries in DBSTYLESPEC<2> must be a record on the Style Definition DBISTYLE file, unless the Field is an MV Grid in which case the list can be javascript, not HTML, style properties. For example use *fontFamily* (javascript) NOT *font-family* (HTML).



The next example demonstrates the use of DBSTYLESPEC as applied to a MV Grid element and a button:

DBSTYLESPEC <1> B.TARGET] DBC.INV.DATE
DBSTYLESPEC <2> style~|border:2px solid green] style~|border:2px solid green
DBSTYLESPEC <3>] R
DBSTYLESPEC <4>] 4
DBSTYLESPEC <5>]

Target Button

	Invoice No	Invoice Date	Invoice Amount
	1	16-01-2017	1.00
	2	16-01-2017	2.00
	3	16-01-2017	3.00
	4	16-01-2017	4.00
	5	16-01-2017	5.00
	6	16-01-2017	6.00
	7	18-01-2017	7.00

The display class in each value position of DBSTYLESPEC<2> may be a list of javascript style properties such as for example:

"style~|border:2px solid red~|styleprop:value"

The *styleprop* must be a javascript style property NOT HTML. As an example use *fontFamily* (javascript) NOT *font-family* (HTML)

Changing the Stylegroup at runtime

DBSTYLEGROUPSPEC

This common variable allows the developer to change the Style Group used for an entire session, alternatively a Style Group can be changed for a single form. DBSTYLEGROUPSPEC can be multivalue delimited to affect multiple Style Groups.

Usage 1: This usage will change the all forms from one Style Group to another
DBSTYLEGROUPSPEC = Stylegroup Name {*VM* Stylegroup Name}
DBSTYLEGROUPSPEC<2> = To Stylegroup {*VM* To Stylegroup}

Eg: DBSTYLEGROUP = “MYDEFAULTGROUP”
DBSTYLEGROUP<2> = “NEWSTYLEGROUP”
In this example, all forms with a Style Group of MYDEFAULTGROUP, will have the Style Group changed to NEWSTYLEGROUP at runtime .

Usage 2: This usage will only change the Style Group for the nominated form
DBSTYLEGROUPSPEC = SCREENROOT|Formname *VM* {SCREENROOT|Formname
DBSTYLEGROUPSPEC<2> = To Stylegroup *VM* {To Stylegroup}

Eg: DBSTYLEGROUP = SCREENROOT : ”|” : DBCLIENT_MYFORM” : VM : “MYDEFAULTGROUP”
DBSTYLEGROUP<2> = “SPECIALGROUP” : VM : “NEWSTYLEGROUP”

In the above example, all forms will have the Style Group changed from MYDEFAULTGROUP to NEWSTYLEGROUP, except for the form DBCLIENT_MYFORM, which will be assigned SPECIALGROUP.

All Style groups referred to in DBSTYLEGROUPSPEC, must be a valid Style group entered in the Style Group Definition form.

Adding Items to a droplist at runtime

DBDROPLISTADD

This option can be used to include additional items to a droplist at runtime. This is useful when data in a file does not match a supporting file. This provides the developer the ability to add any missing codes at runtime. The developer may also provide a description to support the additional items which will alert the user that they should change the selection before any updates.

Usage:

DBDROPLISTADD = Input Field Name
DBDROPLISTADD<2> = Code(s) to add. Separated by value mark.
DBDROPLISTADD<3> = Description(s)
DBDROPLISTADD<4> = Message(s) to help support the code addition. These will be displayed in a message box at runtime.
DBDROPLISTADD<5> = Flag
‘D’ - indicates that the code in <2,pos> is to be deleted
‘R’ – refresh the entire dropdown list (equivalent to re-invoking the form containing the list)

Example:

DBDROPLISTADD = "DBC.ACCOUNT.MANAGER":VM:"DBC.ACCOUNT.MANAGER"
DBDROPLISTADD<2> = "A":VM:"Z"
DBDROPLISTADD<3> = "Xavier":VM:"Zebra"
DBDROPLISTADD<4> = "The account manager Xavier (X) is no longer supported.":VM:"The account manager Zebra (Z) is no longer supported."
DBDROPLISTADD<5> = “”

This option can be used in the AFTER DISPLAY event. In releases after 8.1.3.6 DROPLISTADD can also be used in other events and can be used to amend the **valid input list** of a field property.

It cannot be used to amend the droplist created by setting the Lookup File to a work variable in the form *V:WORK_VARIABLE_NAME*. This is controlled by the developer’s basic code.

Developers should note the following when an input field with a dropdown selection list is updated, via basic code, with a value that is not in the dropdown list. As stated above the common variable DBDROPLISTADD has been implemented to address this very issue but if DBDROPLISTADD is not set by the developer then the browser cannot reflect the new value of the field.

DesignBais Version 6 does not change the dropdown list value even though the invalid value is passed back to the web server:

```
<name>runFunction</name>
<functionParams>select3;FRED</functionParams>
<functionName>g_select</functionName>
```

In the above case a value of “FRED” is passed to a field with a dropdown list that does not contain a code of “FRED”.

In DesignBais Release 8.5.1.1 and later releases invalid values are no longer sent back to the browser. The input element on the form displays a blank value. The value is not null. If it was null then the null option, such as “*Select Type*”, for example, would display.

Although DesignBais no longer returns the invalid value to the browser developers should note that the DBISESSIONS record still holds the wrong value. DBWORK is held in the DBISESSIONS record with id *sessionID*ALL*. This is corrected on the next event that returns the data from the browser so it does not affect any updates.

Developers should be careful not to propagate invalid data across other stored variables.

If the invalid data is required it should be added to the select element using DBDROPLISTADD.

Adding Field Hover Title

DBFIELDTITLE

The new common variable DBFIELDTITLE has been added to make setting field hover title text easy to maintain.The title attribute may also be set in Forms Designer via a Custom Attribute or from code using the DBI.G.AJXCMD function "jqsa".

DBFIELDTITLE<1,nn> = MV list of Field Names
DBFIELDTITLE<2,nn> = Associated list of Field Titles

Passing values (data) to other forms DBPASS.DBVALUE

When a new form is loaded or a sub-form loaded you may want to automatically fill fields on the new form. Some of those fields may have reads and validation processes associated with them.

To pass values from one form to another, or within the same form, there are two variables that can be used:

DBPASS.DBVALUE

Value to be passed (multi-value delimited)

Usage: DBPASS.DBVALUE = *Data Value*

Eg. The developer wishes to populate two fields on the new form. These two fields are part of a multi-part key

DBPASS.DBVALUE = KEY.VALUE1:VM:KEY.VALUE2

DBPASS.DBVALUE.TO

Fields to pass the value to

Usage: DBPASS.DBVALUE.TO = *"Field Name"*

The combination of these two fields can also be used to set defaults on a form.

Eg. The developer wishes to populate two fields on the new form. These two fields are part of a multi-part key

DBPASS.DBVALUE = KEY.VALUE1:VM:KEY.VALUE2
DBPASS.DBVALUE.TO = "KEY.FIELD1.WK":VM:"KEY.FIELD2.WK"

This will populate the two fields on the new form with values of KEY.VALUE1 and KEY.VALUE2

If there are any validation or read events associated with KEY.FIELD1.WK and KEY.FIELD2.WK, they will be processed in the order in which the fields are assigned to DBPASS.DBVALUE.TO

If DBPASS.DBVALUE.TO is left blank the first input field will be filled with the value of DBPASS.DBVALUE. It is recommended that the developer always assign both values to ensure that the correct fields are updated. Tab index changes may change the order of input fields and result in the wrong fields being populated.

Note that if PROCESS.STACK is set then the DBPASS.DBVALUE values are passed to the form that is invoked by PROCESS.STACK.

Note that the DBPASS.DBVALUE function works from the list of values, rather than the list of fields, so it is not possible to pass a null value.

DBPASS.DBVALUE contains too many values

The error "*DesignBais Application Error – An unexpected error has occurred (1)*" displays when DBPASS.DBVALUE is loaded with too many values. The limit may vary but 100 or more values could cuase a problem.

Trapping Modal Form Close Events

DBMODAL.CLOSED.VIA.X

This variable is set to 1 when a modal form is closed via the “X” button.

DBLAYER.CLOSED.VIA.X

This variable is set to 1 when a layered form is closed via the “X” button.

DBRESTORE.BEFORE.MODAL

It is sometimes necessary to trap the close of a modal form. This enables the developer to restore common (if required) as it was before the modal form was closed.

When a modal form is opened, all common variables (including DBSTORE) are saved as a snapshot. When a modal form is closed via the ‘X’ button, the common variable DBMODAL.CLOSED.VIA.X is set to 1.

The developer can then (if required), restore common to the snapshot version by setting the common variable DBRESTORE.BEFORE.MODAL to 1.

Example of usage:

```
IF DBMODAL.CLOSED.VIA.X  THEN
   DBRESTORE.BEFORE.MODAL = 1 ; * Restore the snapshot version of common
END
```

Note that the developer can set the DBRESTORE.BEFORE.MODAL common variable in several places.

If a ‘Process After’ is being used when a button is pressed on a modal form then DBRESTORE.BEFORE.MODAL can be set in this process, with the effect that the snapshot version of common is restored while still in the modal form.

Alternatively DBRESTORE.BEFORE.MODAL can be set in the MODAL RETURN event of the calling form in which case the snapshot version of common is restored after returning to the calling form.

There is only one snapshot version but the timing of the restore may be significant for the developer.

The variable DBRESTORE.BEFORE.MODAL is cleared once it has triggered the restore.

See also the “Subroutine on Main form Close” field in Menu Maintenance.

Moving Fields on a form dynamically

DBMOVEFIELD

This common variable is used to define fields that are to be moved at runtime.

Usage:

```
DBMOVEFIELD<1> = Field to move : VM : Field to Move
DBMOVEFIELD<2> = Xposition to move to : VM: Xposition to move to
DBMOVEFIELD<3> = Yposition to move to : VM: Yposition to move to
```

The X and Y positions are offset automatically by menus if they are being used.

Example Usage:

```
DBMOVEFIELD = "B.NEWACCOUNT":VM:"DBC.CLIENT.NAME":VM:"R.SALES"
DBMOVEFIELD<2> = "400":VM:"500":VM:"400"
DBMOVEFIELD<3> = "600":VM:"200":VM:"200"
```

Moving focus to a position on the form

DBMOVETOFIELD

The variable DBMOVETOFIELD is used to move the form to the location of a field without setting focus to a particular field. This is useful to move the focus to a section on a form without focusing a field within that section

As text fields can now be named, this variable can be used to position the form to a banner field or background field.

Usage:

DBMOVETOFIELD = Field name or DBMOVETOFIELD = Col,Row

Example:

1. DBMOVETOFIELD = "MYFRAME" - MYFRAME can be a text only field
2. DBMOVETOFIELD = “350,400” - will invoke a window.scrollTo(x,y)

Changing the Tab order on a form dynamically

DBTABINDEX

This function will modify the tab index order at runtime.

Usage:

DBTABINDEX<1> = Fields to change : VM : Fields to change
DBTABINDEX<2> = Tab position : VM : Tab position

Moving a section at runtime

DBMOVESECTION

The variable DBMOVESECTION is used to move a section at runtime. DBMOVESECTION identifies the first field in the nominated section and moves all fields in the section by the same adjustment.

Usage:

```
DBMOVESECTION<1> = Section name : VM : Section name
DBMOVESECTION<2> = X pixel shift : VM : X pixel shift
DBMOVESECTION<3> = Y pixel shift : VM : Y pixel shift
DBMOVESECTION<4> = A or blank
```

If “A” the Pixel shift values become absolute position values

Example 1:

Shift the position by 100 X pixels and 200 Y pixels

```
DBMOVESECTION<1,-1> = "Main"
DBMOVESECTION<2,-1> =    100
DBMOVESECTION<3,-1> = 200
```

Example 2:

Move the section to an absolute position.

```
DBMOVESECTION<1,-1> = "Main"
DBMOVESECTION<2,-1> = 10
DBMOVESECTION<3,-1> = 20
DBMOVESECTION<4,-1> = "A"
```

Resizing a field at runtime

DBRESIZEFIELD

The variable DBRESIZEFIELD is used to resize a field or fields at runtime.

Usage:

```
DBRESIZEFIELD<1> = Name of the field : VM : Name of the field
DBRESIZEFIELD<2> = Field Width : VM : Field Width
DBRESIZEFIELD<3> = Field Height : VM: Field Height
DBRESIZEFIELD<4> = Height Adjustment :VM: Height Adjustment
```

Example:

```
DBRESIZEFIELD<1> = "DBC.CLIENT.NAME"
DBRESIZEFIELD<2> = 400
DBRESIZEFIELD<3> = 15
```

The “Name of the field” can be the name of the field on the form, the XML id of the field on the form or the text of a field on the form.

The value in DBRESIZEFIELD is persistent for the current hit. It is therefore necessary to set DBRESIZEFIELD at the point that it is to take effect.

The “Field Width” designates the number of pixels to use for the width. If the value ends with a percentage sign “%” then it will designate that the width is to be that percentage of the form width. Percentage is only valid for certain form elements such as report and image. If the value is greater than 100% then a horizontal scrollbar will appear.

The “Field Height” designates the number of pixels to use for the height.

The DBRESIZEFIELD<4,n> value will adjust the row position of all fields on rows below the field referenced in DBRESIZEFIELD<1,n>.

Adding Video or a Website to your forms

DBADDFRAME

This feature provides the ability to place a website, youtube video, or any other third party site. All DBADDFRAME attributes may contain multi-valued data. If however attribute 8 and 9 have been used as single values the DesignBais engine will now pick up the first multivalue for all subsequent values (in these attributes 8 and 9 only) that are null.

It is important to note, however, that multiple frames on a form will only work if they are from different domains since all frames use the same sessionStorage provided by the browser.

```
DBADDFRAME = A Unique Identifier  - Alpha Numeric characters only.
DBADDFRAME<2> = "D"  or “H” ; Display or Hide.
DBADDFRAME<3> = Site name
DBADDFRAME<4> = Row
DBADDFRAME<5> = Column
DBADDFRAME<6> = Width
DBADDFRAME<7> = Depth
DBADDFRAME<8> = Class name (used to set the “Frame1.className=”)
DBADDFRAME<9> = Zindex number
```

All position and width values are in pixels. Row and Column refer to the position on the form as for any normal form element.

```
DBADDFRAME = "frame1"
DBADDFRAME<2> = "D"
DBADDFRAME<3> = "http://www.DesignBais.com"
DBADDFRAME<4> = 100
DBADDFRAME<5> = 100
DBADDFRAME<6> = 150
DBADDFRAME<7> = 150
DBADDFRAME<8> = “yourStyle”
DBADDFRAME<9> = “zindex number”
```

Once a frame is displayed, it will remain until it is hidden.

Hidden frames requests do not require any positional attributes.

Styling a Multivalue Grid Programmatically

DBMVPROP

The common variable DBMVPROP is used to format mv grids at run time.
This variable has 26 attributes with each multivalue representing a different grid.

<1> Screenroot : “|” : Grid Name
<2> Overwrite the standard box height (grid depth)
<3> Exterior class of the box (MV Border Class)
<4> Change the default header height
<5> Change the grid header class (MV Header Class)
<6> Change the default Header Mouseover Class
<7> Change the class in the top left corner (Top Left Class)
<8> Change the class of the add button (Add Button Class)
<9> Change the class of the delete button (Delete Button Class)
<10> Change the class of the insert button (Insert Button Class)
<11> Set to 1 to suppress the scrollbar, set to 0 to enable the scrollbar
<12> Class to invoke on the grid row that has focus (Row Class on Focus)
<13> Class to invoke when the grid row loses focus (Row Class on Focus Out)
<14> Text to replace the standard add row text which is a ‘+’ sign (overridden by any add button class)
<15> Text to replace the standard delete row text which is an ‘x’ character (overridden by any delete button class)
<16> Text to replace the standard insert row text which is a ‘>’ symbol (overridden by any insert button class)
<17> Text to replace the standard top left corner text which is a space (overridden by any top left class)
<18> Allow insert (Y (yes), N (no), A (add only), I (insert only))
<19> Allow delete (Y (yes), N (no), D (with dialog))
<20> MV Grid add button position
Null (top left) R (top right) BL (bottom left) BC (bottom center) BR (bottom right)

<21> MV Grid delete button position
Null (left) R (right)
<22> MV Grid insert button position
Null (left) R (right)
<23 MV Grid Footer Height in pixels
<24> MV Grid Footer Class
<25> MV Grid Detail Class
<26> MV Grid Force refresh - Set to 1 or Y to force the refresh of the grid, otherwise leave null
<27> MV Grid column header attributes pipe separated

```
IF DBMVPROP<20,MVPOS> # "" THEN
   DBMVADDPOS = DBMVPROP<20,MVPOS>
   VALS = CHANGE('R,BL,BC,BR',",",VM)
   LOCATE DBMVADDPOS IN VALS<1>,1 SETTING DUM ELSE DBMVADDPOS = ''
END
IF DBMVPROP<21,MVPOS> # "" THEN
   DBMVDELPOS = DBMVPROP<21,MVPOS>
   IF DBMVDELPOS # 'R' THEN DBMVDELPOS = ''
END
IF DBMVPROP<22,MVPOS> # "" THEN
   DBMVINSPOS = DBMVPROP<22,MVPOS>
   IF DBMVINSPOS # 'R' THEN DBMVINSPOS = ''
END
*
DBMVFOOTERHEIGHT = DBMVPROP<23,MVPOS>
DBMVFOOTERCLASS = DBMVPROP<24,MVPOS>
DBMVDETAILCLASS = DBMVPROP<25,MVPOS>
```

Developers should note several points from the code example below:

- In the code snip below note that a work field is assigned to DBVALUE at the start of the validation. This is commonly done to allow code that processes the work variable to be used in the validation event. If the value of the work variable is changed by this validation event processing then it is important to set DBVALUE to the changed value since, when the validation is finished, DesignBais sets the eventsource field to the value in DBVALUE.

- You can use DBMVPROP<26> = 1 to force DesignBais to refresh the grid.

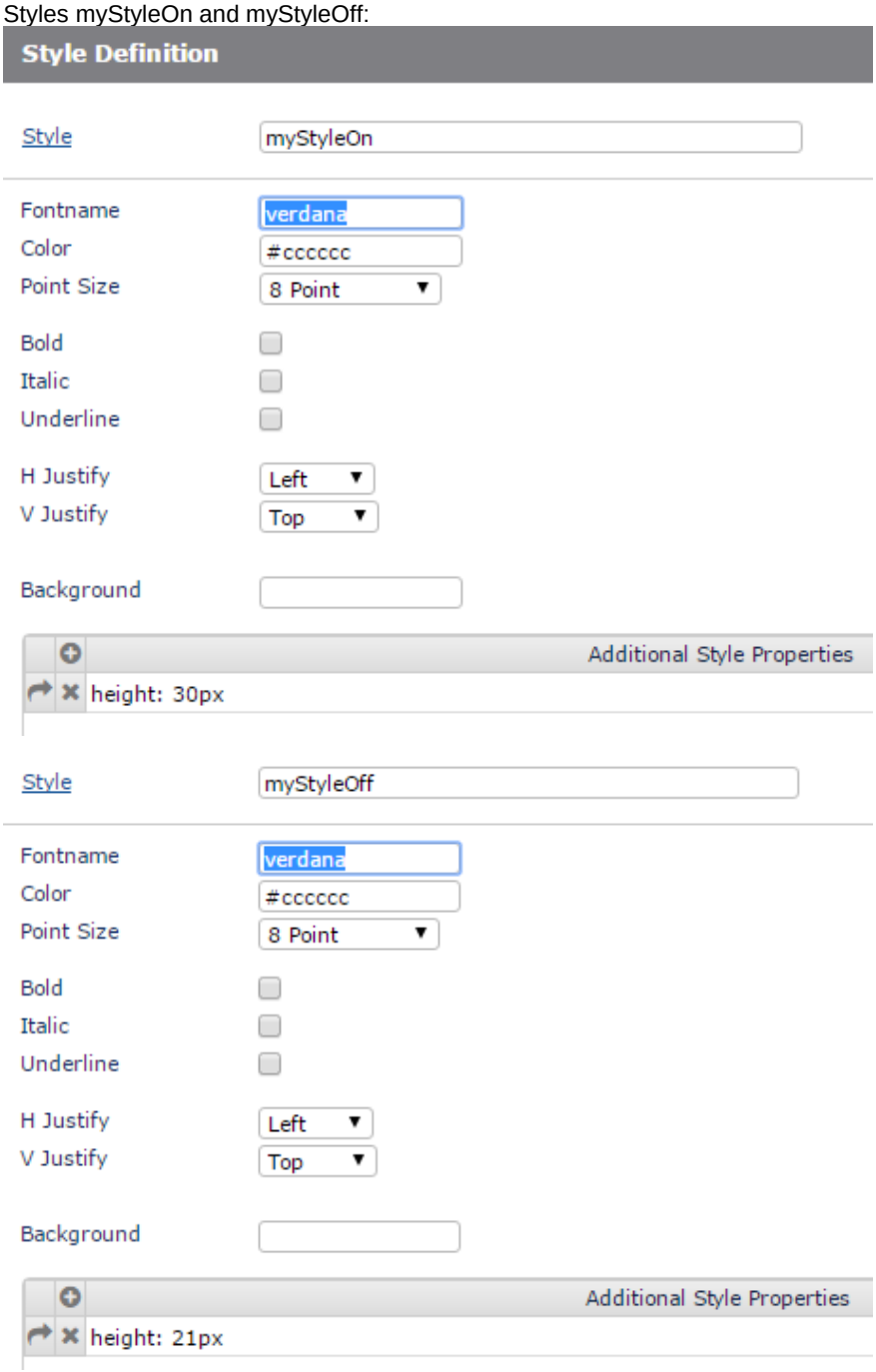
```
CASE SCREEN.NO = "STYLESPEC" AND EVENTSOURCE = "DEM.WORK19.WK"
   DBWORK<DEM.WORK19.WK,DBMVCOUNT> = DBVALUE
   SMAX = DCOUNT(DBWORK<DEM.WORK19.WK>,VM)
   FOR JJ = SMAX TO 1 STEP -1
      IF DBWORK<DEM.WORK19.WK,JJ> = '' THEN
         DEL DBWORK<DEM.WORK19.WK,JJ>
      END
   NEXT JJ
   SMAX = DCOUNT(DBWORK<DEM.WORK19.WK>,VM)
   IF SMAX = 0 THEN
      DBDS<-1> = 'This field requires at least 1 row of data'
      DBRETURN.TO.FIELD = 'DEM.WORK19.WK~|1'
   END
   DBVALUE = DBWORK<DEM.WORK19.WK,DBMVCOUNT>
   DBMVPROP = SCREENROOT:"|ASSOC2"
   DBMVPROP<26> = 1
```

Example:

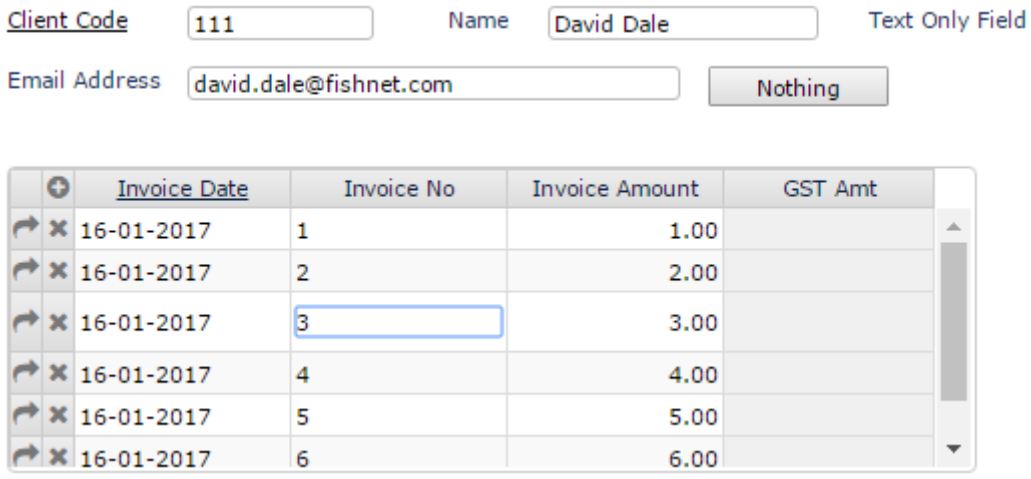
The following Basic code sets the class to be invoked in a particular MV grid on form GENERALTEST

```
AFTER.DISPLAY:     * Process After Display
   BEGIN CASE
      CASE SCREEN.NO="GENERALTEST"
         DBMVPROP = ""
         DBMVPROP<1> = SCREENROOT:"|":"INVLIST1"
         DBMVPROP<11> = ""
         DBMVPROP<12> = 'myStyleOn'
         DBMVPROP<13> = 'myStyleOff'
```

Styles myStyleOn and myStyleOff:



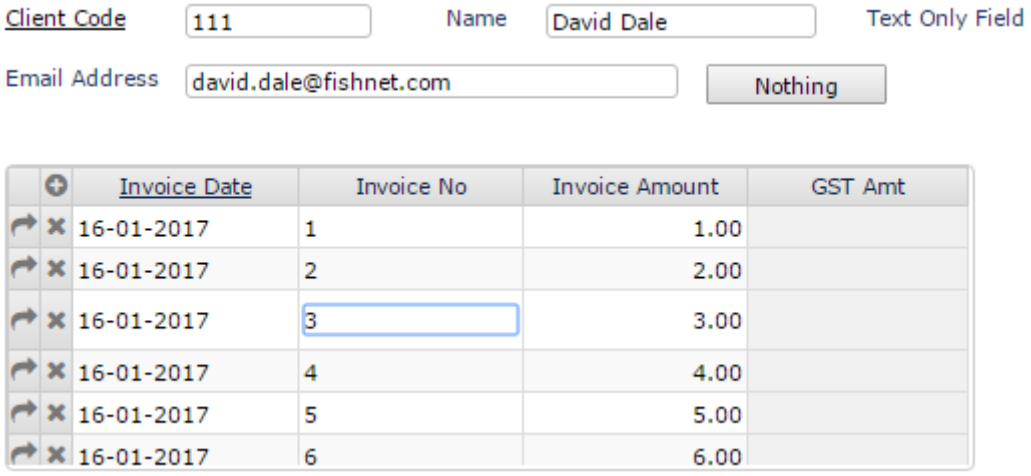
Running the form and setting focus on row 3 of the MV grid INVLIST1 demonstrates the effect whereby the row with focus has a height of 30 pixels. Rows without focus return to the standard 21 pixel height.



Changing the Basic code to have:

```
DBMVPROP<11> = 1
```

causes the scrollbar to be suppressed:



Note that when testing a form in Forms Designer DBMVPROP requires SCREENROOT to have *.DESIGNER* appended.

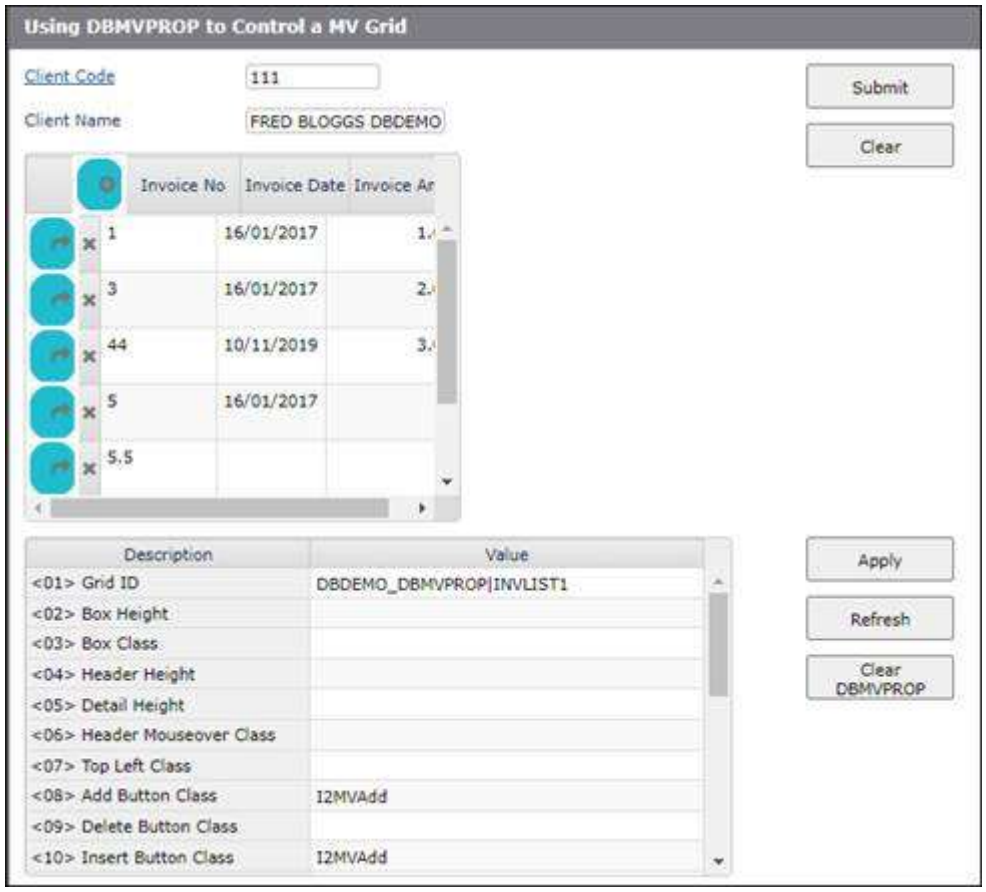
Example DBMVPROP<27>:

Set up mouseover title attribute for column headers in a multivalue grid.

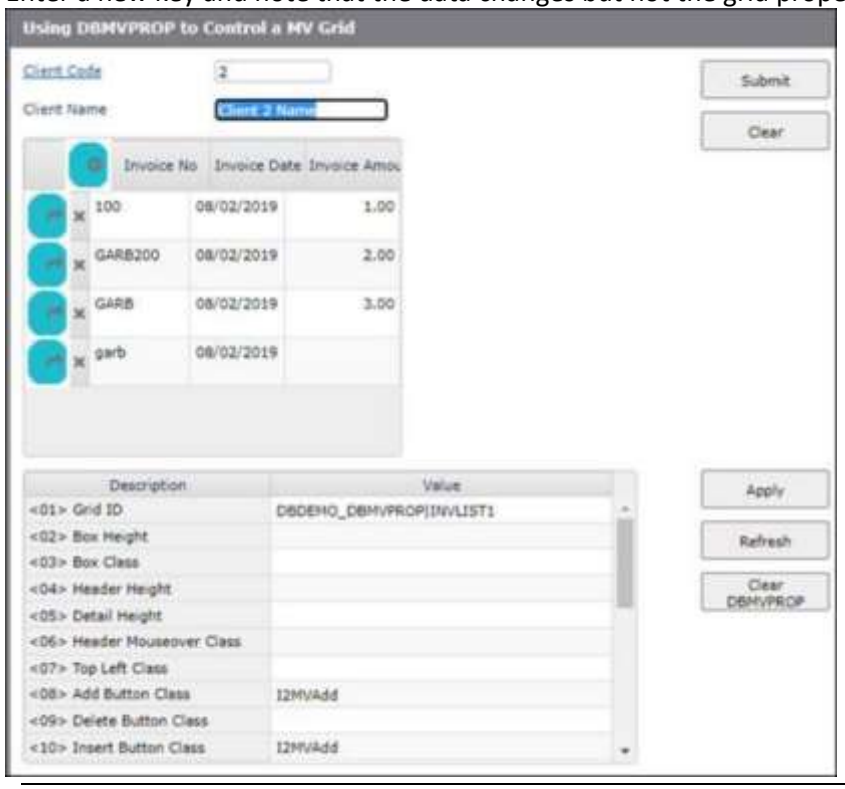
```
           LOCATE SCREENROOT:"|MV1" IN DBMVPROP<1>,1 SETTING MVPOS ELSE
              INS SCREENROOT:"|MV1" BEFORE DBMVPROP<1,MVPOS>
           END
           DBMVPROP<27,MVPOS>   = ''
           DBMVPROP<27,MVPOS,2> = 'title="Click to sort by file, file type and item. Duplicates are removed."'
           DBMVPROP<27,MVPOS,4> = 'title="Click to open all basic subroutine items on this checklist page in code editor.&#013;(i.e.
items on files listed in the Basic Library File list in your user record)"'
           DBMVPROP<27,MVPOS,5> = 'title="Click to select an item from the nominated file."'
           DBMVPROP<27,MVPOS,6> = 'title="Click to sort by change type, file, file type and item."'
           DBMVPROP<27,MVPOS>   = CHANGE(DBMVPROP<27,MVPOS>,SVM,"|")
```

Note the use of
 to force a breakline.

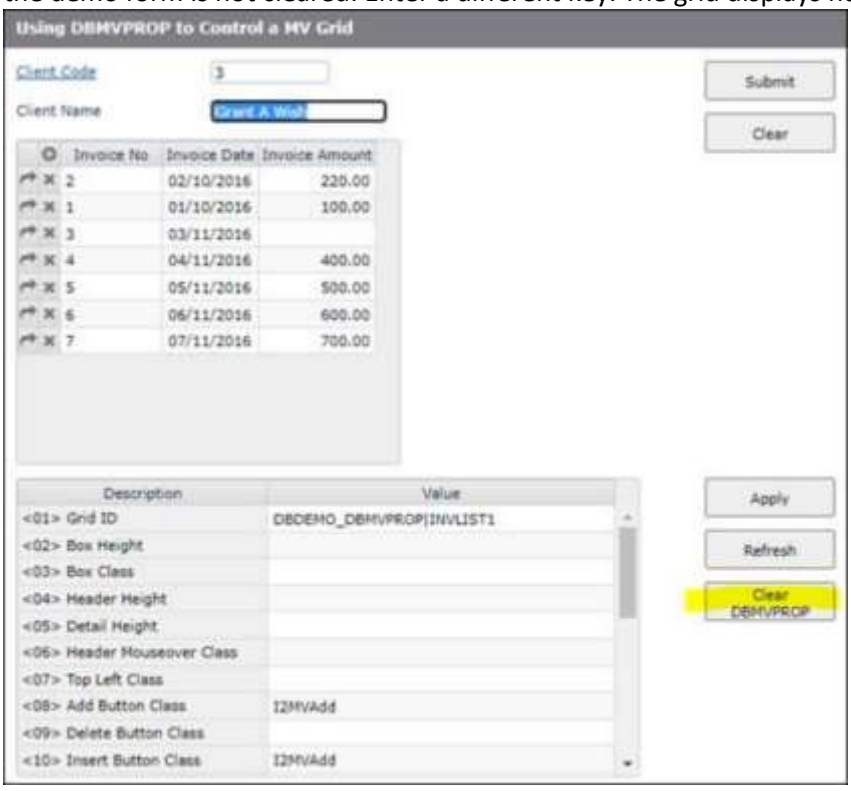
Example using the DesignBais Demo form. Enter a new *Add* and *Insert* button class then click *Apply*.



Enter a new key and note that the data changes but not the grid properties. They are derived from DBMVPROP.



Click the *Clear DBMVPROP* button. This clears the common variable DBMVPROP. The form work variable at the base of the demo form is not cleared. Enter a different key. The grid displays normally again since DBMVPROP is null.



Multivalue Grid Cell Focus

DBLASTMVCELL

Holds details of the last multivalue grid cell to gain focus. If the user tabs into a grid cell or clicks a cell then DBLASTMVCELL is set

Form Group Name	DBLASTMVCELL<1,1> = 'MV1'
Form field XML label id	DBLASTMVCELL<2,1> = 'text97v1'
Grid row	DBLASTMVCELL<3,1> = '1'
Grid column	DBLASTMVCELL<4,1> = '5'

Multivalue Selection

DBSELECTKEY

DBSELECTKEY is VM delimited.

DBSELECTKEY<1,1> The name of the multivalued field to be populated. This is set by the developer.

DBSELECTKEY<1,2> Sub-valued list of selected keys that are “pushed” into the multivalued field – returns the selected items.

DBSELECTKEY<1,3> Holds the value of DBMVCOUNT to commence from. This is set by the developer.

DBSELECTKEY<1,4> Holds “INSERT” or “REPLACE” to indicate whether selected keys are inserted at the DBMVCOUNT position or whether selected keys replace all currently selected keys.

DBSELECTKEY<1,5> Holds the Form Group Name for the multivalue grid. This is set and used by DesignBais.

DBSELECTKEY<1,6> Holds the first MV position to be deleted when the new list is shorter than the current list. This is set and used by DesignBais.

DBSELECTKEY<1,7> Holds the number of original keys (set by DesignBais and used in DBI.G.MVW3CNET when rebuilding the grid)

DBSELECTKEY<1,8> Reserved for application use. For example it could hold a “CANCELLED” flag. This is not used by DesignBais.

Clear DBSELECTKEY to cancel the process.

Developers can reference the demo forms DBDEMO_DBSELECTKEY and DBDEMO_DBSELECTKEY.EXEC which demonstrate how this common variable can be used.

Draggable Panel

DBDRAG

Common variable used for draggable panels.

Refer to the demo form DBDEMO*DRAGPANEL.

The DesignBais subroutine DBI.G.PANEL is used to control the display of draggable panels.

```
SUBROUTINE DBI.G.PANEL(DPREC)
* Description: Routine to control display of Draggable Panel for a form
*
* All fields in a section with an Initial State of DRAG - Draggable are placed into a dbmasterdivnn where nn = form
DBWLEVEL+30
* DBDRAG is set in DBI.G.XMLNET as the fields are extracted from the base form.
*     DBDRAG<01,nn> = TSCREENROOT:"@":FIELD.SECTION:"@":DBWLEVEL which controls the first 10 associated attributes
*     DBDRAG<02,nn> = First Row in the section
*     DBDRAG<03,nn> = First Column in the section
*     DBDRAG<04,nn> = Last Row in the section
*     DBDRAG<05,nn> = Last Column in the section
*     DBDRAG<06,nn> = Section Name
*     DBDRAG<07,nn> = Widest Column Span in Section
*     DBDRAG<08,nn> = reserved for future use
*     DBDRAG<09,nn> = reserved for future use - current state
*     DBDRAG<10,nn> = reserved for future use - new state
*
*     DBDRAG<11,vv> = reserved for future use - vv = DBWLEVEL
*     DBDRAG<12,vv> = First Row in the panel
*     DBDRAG<13,vv> = First Column in the panel
*     DBDRAG<14,vv> = Last Row in the panel
*     DBDRAG<15,vv> = Last Column in the panel
*     DBDRAG<16,vv> = Drag handle (dbaisModalTitle) height - needed to size
*     DBDRAG<17,vv> = reserved for future use
*     DBDRAG<18,vv> = reserved for future use
*     DBDRAG<19,vv> = DRAG panel current state - E=enabled, H=Hidden, Initially empty
*     DBDRAG<20,vv> = DRAG panel new state - E=enabled, H=Hidden
*
* DPREC provides the parameters to control the panel
*     DPREC<01>     = Panel State - E=enabled, H=Hidden
*     DPREC<02>     = Panel width - numeric - px. Use "RESET" to return to the original dimensions
*     DPREC<03>     = Panel depth - including drag handle - numeric - px
*     DPREC<04>     = Panel Move Type - 0 = Offset from click position or 1 = No Offset
*     DPREC<05>     = Panel X-Pos Move
*     DPREC<06>     = Panel Y-Pos Move
```

Two Pass Report Calculations

DBBREAKTOTALS

Every name of an accumulated field on a report is referred to in a field named DBACCUMULATORS.LIST. Totals for the current break levels are stored in the variable named **DBBREAKTOTALS.** This variable is attribute-delimited for each break level on the report. Each attribute is value delimited for each of the accumulated totals.

.

Eg:

Client Code	Name	Month Sales	Year Sales
C00001	Max Monk	100.00	500.00
C00002	Harry Monk	100.00	400.00
Sales Executive	**Barry Miner**	**200.00**	**900.00**
C00003	George Basin	100.00	400.00
C00004	Margaret River	100.00	400.00
Sales Executive	**Sally Myers**	**200.00**	**800.00**
Report Total		**400.00**	**1700.00**

At all points in the second pass, DBACCUMULATORS.LIST =]]MTD.SALES]YTD.SALES
(']' being @VM)

This enables the developer to locate the required total in the DBBREAKTOTALS variable

Example of Usage:

```
LOCATE "MTD.SALES" IN DBACCUMULATORS.LIST<1> SETTING POS THEN
        EXEC.MTD  = DBBREAKTOTALS<1,POS> ;* Value associated with the Sales Executive Break
        REP.TOT   = DBBREAKTOTALS<2,POS> ;* Value associated with the Report Total.
        DERIVED.1 = (DBRECORD<MTD.SALES> / EXEC.MTD) * 100  ; * First % Column
        DERIVED.2 = (DBRECORD<MTD.SALES> / REP.TOT) * 100 ; * Second % Column
END ELSE
        DERIVED.1 = 0
        DERIVED.2 = 0
END
```

At this point we now have the Executive total and the Report total. It is then possible to establish what the percentage of each total the client "C00001" [line 1] contributes to both the Executive total and the Report Total. These amounts could then be added to the as follows:

Client Code	Name	Month Sales	%	Year Sales	%
C00001	Max Monk	100.00	50	500.00	55
C00002	Harry Monk	100.00	50	400.00	45
Sales Executive	**Barry Miner**	**200.00**	**50**	**900.00**	**53**
C00003	George Basin	100.00	50	400.00	50
C00004	Margaret River	100.00	50	400.00	50
Sales Executive	**Sally Myers**	**200.00**	**50**	**800.00**	**47**
Report Total		**400.00**	**100**	**1700.00**	**100**

Break % totals are accumulated during the subroutine call in break fields.

Suppressing Header and Footer Sections on a Report

DBSUPPRESSREPORTSECTION

In some instances you may wish to suppress a Header or a footer section during a report, particularly in Grand Total breaks.

There are three settings for this variable and the setting can be multivalue delimited. The settings are HEADER, COLHEADER and FOOTER.

Usage: DBSUPPRESSREPORTSECTION = Section to Suppress *VM* {Section to Suppress}

Eg: DBSUPPRESSREPORTSECTION = “HEADER”:VM:”FOOTER”

In the above example the HEADER and FOOTER sections would be disabled.

The Last Record on a Report

DBREPORTLASTRECORD

This variable is set to one “1” at the last record in the report. This provides the opportunity to track the last record in a report if there are any special programming requirements

Unique Session Id for Reports

DBREPORT.SESSION.ID

Some applications perform aggregation of data before running reports. This is usually performed by creating a temporary file and defining DBWITCHGROUP and DBSWITCHFILE to point the report generator at the temporary file. The temporary file is usually created with SESSION.ID as part of the filename. The issue with this is that if the user runs multiple reports concurrently, this file may be cleared or overwritten. DBREPORT.SESSION.ID can be used in place of SESSION.ID for this purpose. DBREPORT.SESSION.ID will always be unique and is formed from the session id concatenated with the time.

Determining the pass number in a two pass report

DBBREAKPASS

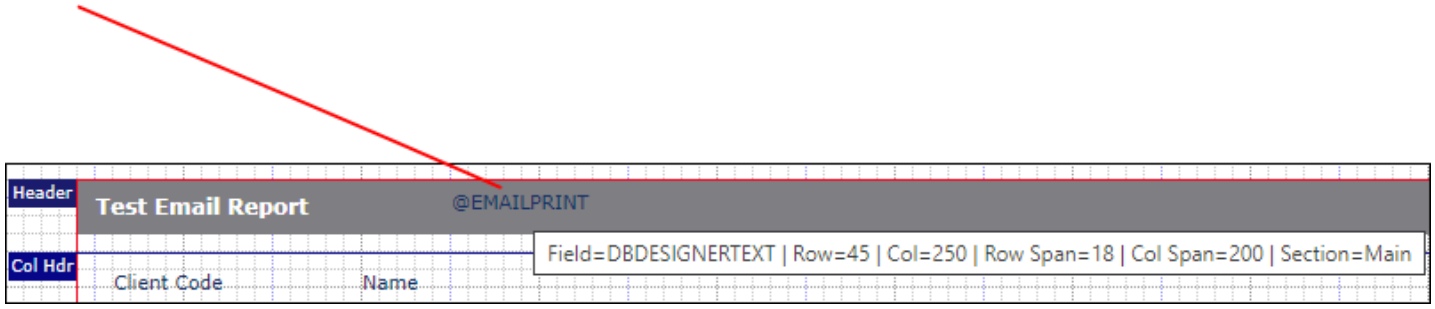
This variable can be used in the “REPORT READ” slot in a report to determine the current pass number in a DesignBais report. It is set to 1 for the first pass and zero for the second pass.

Creating an Email Distribution in a report

DBEMAILTO

@EMAILPRINT

In the top line of a DesignBais report you may enter the @EMAILPRINT tag by adding a Text Only field containing '@EMAILPRINT' as the text value. This text field must be placed within the report header as it must be uniquely identified on each report page.



A complete email tag will have a valid email address concatenated by a pipe character after the @EMAILPRINT text field on the report. For example a page of the report could have "@EMAILPRINT|person@domain.com".

The email tag is completed when the DBEMAILTO is set to a non-null value. This is normally done in the "REPORT READ" or "AFTER READ" slots in a DesignBais report.

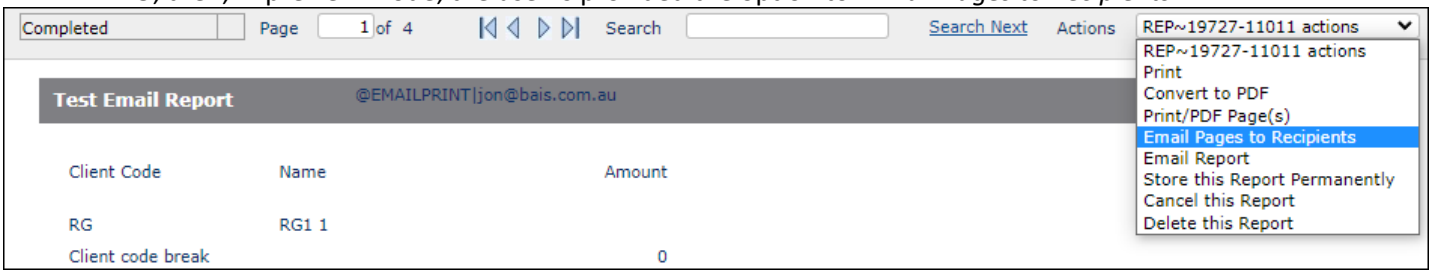
Eg.

```
AFTER.READ:
    BEGIN CASE
        CASE EVENTSOURCE = "STATEMENTS"
            DBEMAILTO = DBRECORD<DBC.EMAIL.ADDRESS> [; another email address]
```

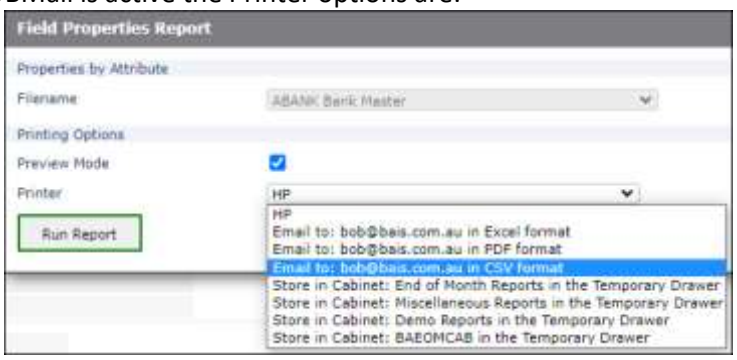
On return to the report generator the @EMAILPRINT tag will be concatenated to the value in DBEMAILTO. Multiple email addresses can be assigned to DBEMAILTO separated by a semicolons.

It is therefore possible in the AFTER READ slot of a report to, for example, read a record, extract an email address from that record and assign DBEMAILTO = "email address". If the @EMAILPRINT tag is in the header of the report then the value of DBEMAILTO will be assigned to that page of the report. Thus each page of the report can contain a different email address thus allowing each page of the report to be emailed to the appropriate email address.

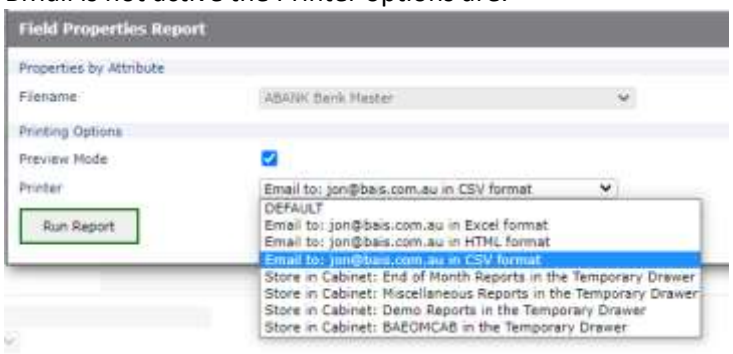
When a report is run and email recipient preferences have been established using @EMAILPRINT and the common variable DBEMAILTO, then, in preview mode, the user is provided the option to "*Email Pages to Recipients*":



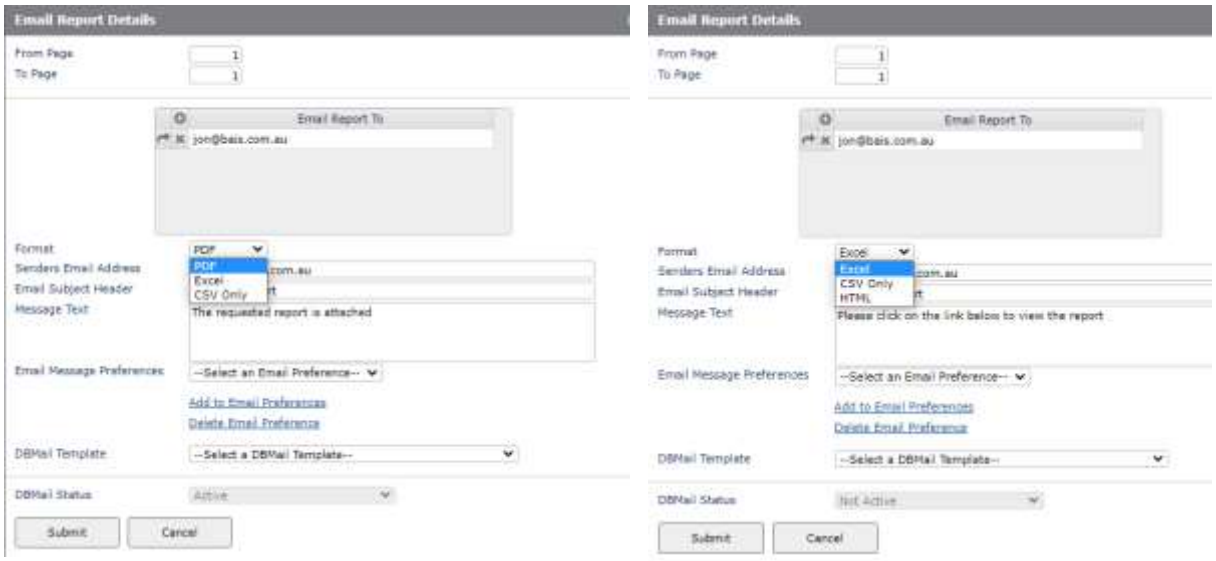
When DBMail is active the Printer options are:



When DBMail is not active the Printer options are:



Selecting the *Email Pages to Recipients* option opens the *Email Report Details* form. On the left the dropdown reflects that DBMail is active. On the right DBMail is inactive.



You can then select from the Email Message Preferences dropdown. Selecting one of the dropdown preferences retrieves a set of saved attributes:

- Senders Email Address
- Email Subject Header
- Message Text

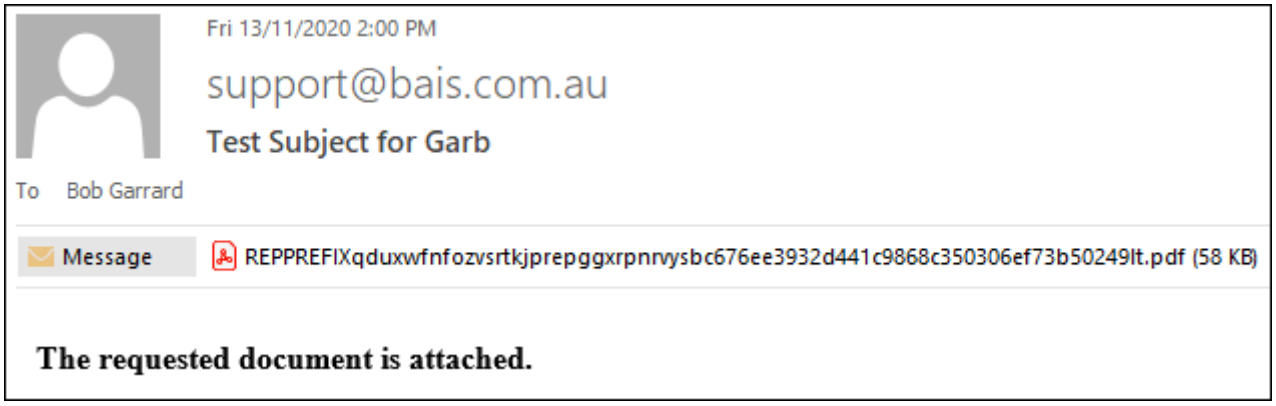
A set of attributes can be saved for each login id. A user is able to maintain and use the set of attributes that belong to his/her user id.

Attributes belonging to other users can be selected but not changed. Use the '*Add to Email Preferences*' to add or update the preferences for your login id. Use '*Delete Email Preference*' to remove your set of preferences.

Note that the report as displayed on the screen shows the *@EMAILPRINT*| tag followed by the email address. This tag is not included in the actual report that is emailed to the recipient.

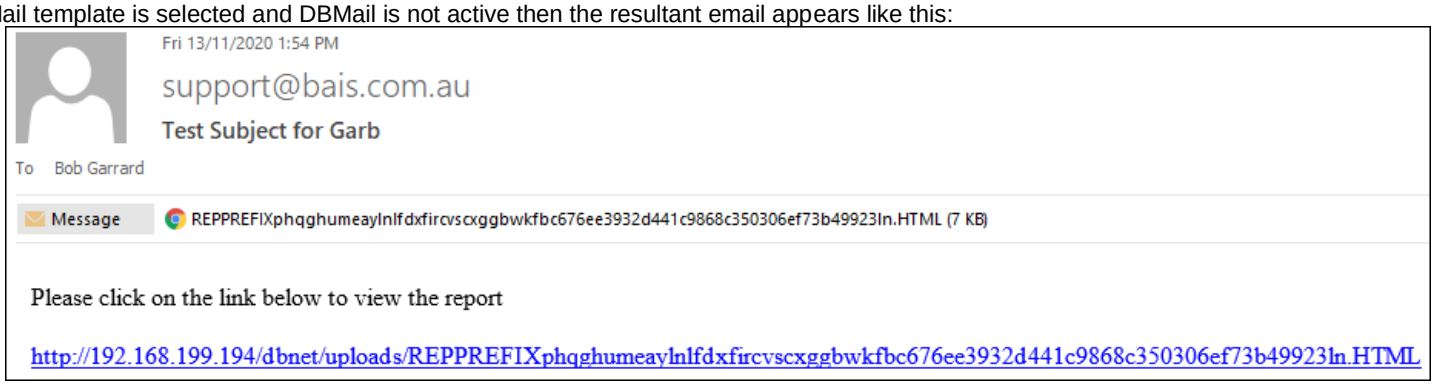
In a report break routine setting *DBEMAILTO="NOEMAIL"* will remove any email addresses from the break page. This allows you to email report pages to the assigned email address but pages containing break totals will not be emailed.

If a DBMail template is selected and DBMail is active then the resultant email appears like this:



The body of this email is derived from the template and the report is attached as a PDF file. The developer can create DBMail Templates that utilise HTML features:

If a DBMail template is selected and DBMail is not active then the resultant email appears like this:



Changing the Default Email From Address

DBEMAILFROM

The email from address in system generated emails in derived in the following manner:

- If DBEMAILFROM is not null then this is used,
- or if the System Parameters record has a default Email From address, then this is used,
- or the email address of the DesignBais user record is used.

Changing Form Content at Runtime

DBFORMLOCAL

This common variable is used to allow the developer to make changes to a base form before execution.

When this variable is set to a form name DesignBais expects the form to be written to the DBISESSIONS file.

The format of the write should be SESSION.ID:"_":FILENAME:"_":FORMNAME.

DBFORMLOCAL is then set by the developer (at runtime) to the form name to get locally from the sessions file (DBISESSIONS)

Usage: DBFORMLOCAL = FILENAME_FORMNAME *VM* {FILENAME_FORMNAME}

Eg: DBFORMLOCAL = DBCLIENT_TESTFORM

Setting this changes DesignBais form-load behaviour

Typically the DesignBais form load process is:

Read the form from DBIFORMS file otherwise, read the form from the DBISYSFORMS file.

Setting the DBFORMLOCAL to a form or forms (value mark delimited) will add a read from the DBISESSIONS file.

Eg:

```
READ FORMDATA FROM F.DBISESSIONS,SESSION.ID:"_":FILENAME:"_":FORM.NAME ELSE
   READ FORMDATA FROM F.DBIFORMS,FILENAME:"_":FORMNAME ELSE
      READ FORMDATA FROM F.DBISYSFORMS,FILENAME:"_":FORMNAME
```

This provides the developer with an extra level of flexibility in the form management process. The developer can programmatically modify an existing form, write it to the DBISESSIONS file, include the Formname in DBFORMLOCAL and the form is replaced at runtime.

When the session is closed by the user, any forms loaded as a result of DBFORMLOCAL will be removed from the DBISESSIONS file.

Fields on the modified form should be resequenced using the DBI.G.RESEQ.FORMDATA routine. This will sort the form fields by the tab index and will reset the XML labels.

When returning from a form that has been called using DBFORMLOCAL it is best to remove the Form Id from DBFORMLOCAL. This can be done in the MODAL RETURN event. Remember that DBFORMLOCAL is multivalued so locate the Form Id and remove it from the list.

Controlling Sections At Runtime

DBSECTIONSPEC

This variable is used to control the display characteristics of a section at runtime.

Usage:

DBSECTIONSPEC = Section Name *VM* {Section Name}
DBSECTIONSPEC<2> = Change to State

The valid Parameters for the Change to State are:

H	Hidden
E	Enabled
D	Disabled
C	Collapse
HE	Hidden on report and on excel export file

Eg:

DBSECTIONSPEC<1> = "Section1":VM:"Address3":VM:"Address2"
DBSECTIONSPEC<2> = "C":VM:"H":VM:"E"

The Section Screen of the forms designer must still be used to set the initial state of all sections. From this point all sections can be controlled programmatically if required. Note that Text type fields within a section that is disabled cannot be enabled.

Note that DBSECTIONSPEC is cleared after every hit. The developer should therefore append values to it as required. These will take effect at the next hit after which the variable is cleared by DesignBais.

DBENABLEFIELD

The common variable DBENABLEFIELD is used to enable, disable and hide a field or control on a form programmatically. The 4th attribute applies to multivalue grids and allows specific rows to be disabled or hidden. Leave blank if all rows are to be effected.

At release 7.2.2 two new options were introduced. Use the DBENABLEFIELD option 'U' to redisplay a MV column compressed or hidden by DBENABLEFIELD = "C" or "H". The uncompress option will not affect the current readonly or disabled attributes of the column.

Usage:

DBENABLEFIELD<1> = Screen Name : VM : Screen Name
DBENABLEFIELD<2> = Field : SVM : Field : VM Field
DBENABLEFIELD<3> = D/E/H/C/U : SVM : D/E/H/C/U : VM D/E/H/C/U
DBENABLEFIELD<4> = MV Row : SVM : MV Row: VM MV Row
(D/E/H – Disable, Enable, Hidden)

Example:

DBENABLEFIELD<1> = ‘DBCLIENT_D1’ : VM : ‘DBCLIENT_D2’
DBENABLEFIELD<2> = ‘B.SUBMIT’ : SVM : ‘B.DELETE’ : VM ‘B.SUBMIT’
DBENABLEFIELD<3> = ‘D’ : SVM : ‘D’ : VM ‘H’
DBENABLEFIELD<4> = 3 : SVM : 5 : VM 1

Once set within your basic code this variable remains set. Your code must clear it in order to remove its influence.

On every server hit DesignBais compares the contents of DBENABLEFIELD to the value as at the previous hit. If there is any difference then the settings in DBENABLEFIELD are acted on. It is recommended that the developer sets DBENABLEFIELD by using the following structure where FLDNAME is the field to be enabled, disabled or hidden:

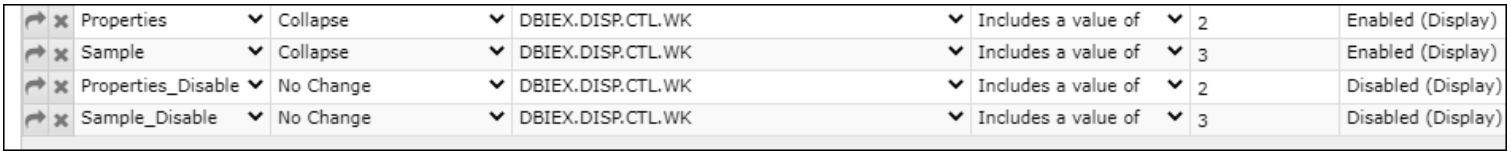
```
LOCATE SCREENROOT IN DBENABLEFIELD<1> SETTING SPOS ELSE;* locate the value position
      INS SCREENROOT BEFORE DBENABLEFIELD<1,SPOS>
      INS '' BEFORE DBENABLEFIELD<2,SPOS>
      INS '' BEFORE DBENABLEFIELD<3,SPOS>
      INS '' BEFORE DBENABLEFIELD<4,SPOS>
END
LOCATE FLDNAME IN DBENABLEFIELD<2,SPOS> SETTING FPOS ELSE;* locate the subvalue position
        INS FLDNAME BEFORE DBENABLEFIELD<2,SPOS,FPOS>
        INS '' BEFORE DBENABLEFIELD<3,SPOS,FPOS>
        INS '' BEFORE DBENABLEFIELD<4,SPOS,FPOS>
END
DBENABLEFIELD<3,SPOS,FPOS> = 'D';* disable the field (use E to enable, or H to hide)
```

Note that when sections are collapsed by DesignBais the state of all the collapsed fields is set to 'H'. Therefore the developer must take this fact into account when setting DBENABLEFIELD. If the developer changes the state of a collapsed field then that field will reappear on the form even though the Section Control 'Value for Condition' setting has not changed and may still indicate that the section should be collapsed.

The following example demonstrates a method that allows the developer to disable a field in a collapsing section.

A section that, under a certain condition, is collapsed may contain a field that is always to be disabled.
Setting the *Field Disabled* on the form field definition will not keep the field disabled because the uncollapsing of the section will override the disabled property.

To overcome this assign the field or fields to be disabled to a subsection. In the image below the subsections are *Properties_Disabled* and *Sample_Disabled*.



Properties	Collapse	DBIEX.DISP.CTL.WK	Includes a value of	2	Enabled (Display)
Sample	Collapse	DBIEX.DISP.CTL.WK	Includes a value of	3	Enabled (Display)
Properties_Disable	No Change	DBIEX.DISP.CTL.WK	Includes a value of	2	Disabled (Display)
Sample_Disable	No Change	DBIEX.DISP.CTL.WK	Includes a value of	3	Disabled (Display)

Set the *Initial State* of the subsection(s) to *No Change*.
Assign the same *Condition by Field, Condition Operand,* and *Value for Condition* as the section to which the subsection belongs.
Set the *Section State* to *Disabled (Display)*.

Raising a Button Click Event

DBBUTTONCLICK

This variable is used to raise a button-click event at runtime. This simulates the user pressing the defined button. This is extremely useful for automatically invoking a search form if data is entered in a name field and the key field is blank.

This event cannot be used recursively, i.e.: one DBBUTTONCLICK event cannot spawn another, and so on.

Usage:

DBBUTTONCLICK = Name of the button

Example:

DBBUTTONCLICK = “B.SEARCH”

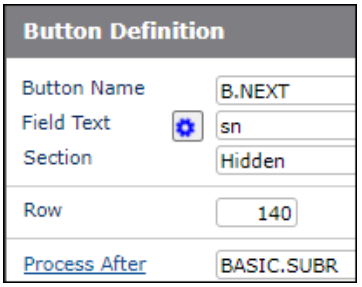
Raising a button-click event from a text field can be accomplished as follows.

In this example the form is on file CUST. Add a hidden button to the form, called for example B.NEXT.

Add a text field adjacent to the hidden button containing the following Field Text

```
<i title='Go to Next Value' class='fa fa-forward'
onclick="document.getElementsByName('CustBNext')[0].click();" aria-hidden='true'
style='color:#777;font-size:13px;'></i>
```

In the image to the right the button is marked as “sn” and is hidden. The text field displays the awesome font character *fa-forward*.



At run time the field displays like this:

Clicking the text field actions the process after subroutine BASIC.SUBR.

Javascript Commands

DBAJAXCMD

If the common variable DBAJAXCMD is populated then its contents will be added to the HTML output string that is sent to the browser.

Developers may populate this variable in their basic code. Each command should added as a new attribute. All commands must be terminated with “; “ character string (semicolon followed by 3 spaces).

DBAJAXCMD is useful to add your own javascript or jQuery commands for any HTML elements that you embed in your forms.

For form elements known to DesignBais please use the existing recommended methods.

It is recommended that developers use the functions in the subroutine DBI.G.AJXCMD.

Refer to the DesignBais Subroutines section of this manual.

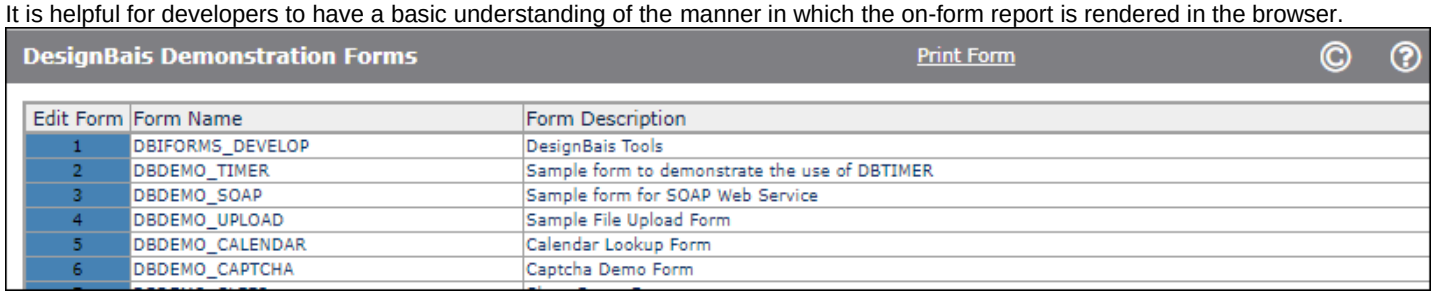
On-form Reports

A DesignBais form can have up to ten On-form Reports displayed. Each report is given a unique report number between one and ten. The developer has complete control over the look and feel of the On-form Report.

Below is an example of an On-form Report and the BASIC routine used to create it.

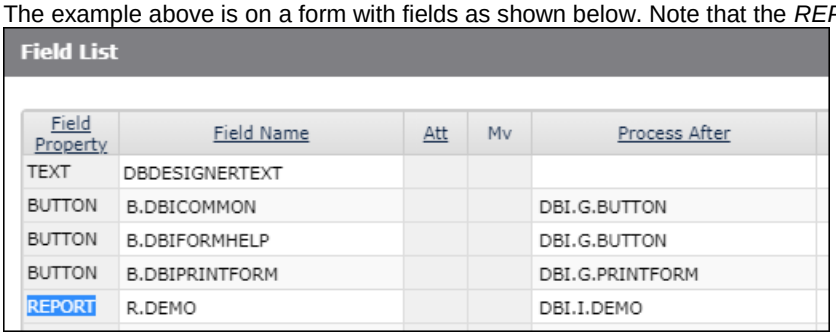
On-form Report example

It is helpful for developers to have a basic understanding of the manner in which the on-form report is rendered in the browser.



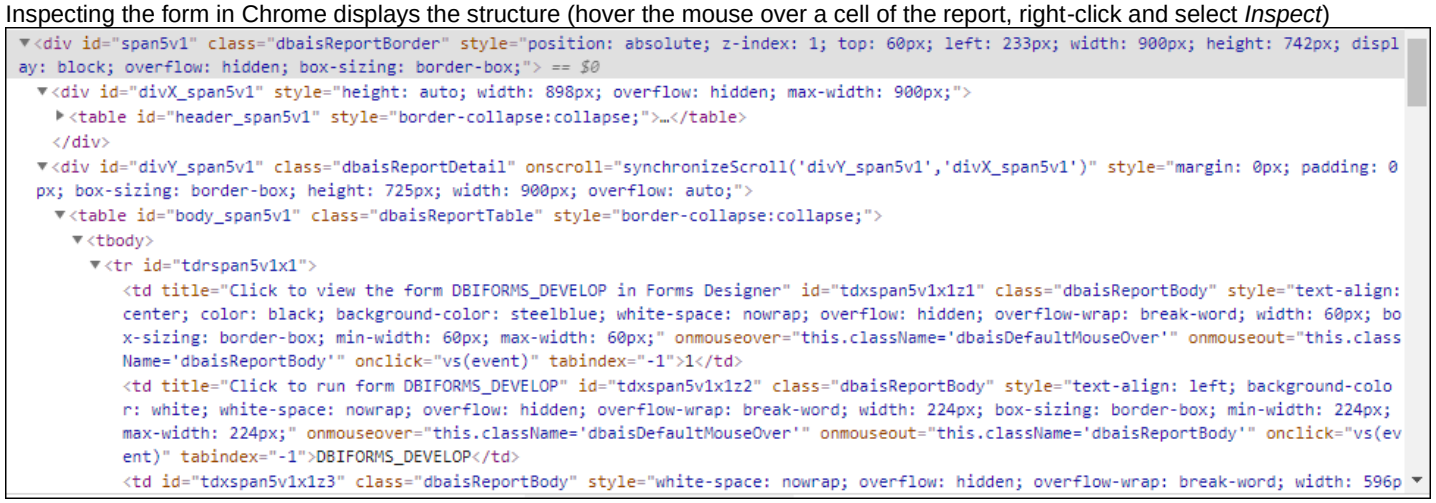
Edit Form	Form Name	Form Description
1	DBIFORMS_DEVELOP	DesignBais Tools
2	DBDEMO_TIMER	Sample form to demonstrate the use of DBTIMER
3	DBDEMO_SOAP	Sample form for SOAP Web Service
4	DBDEMO_UPLOAD	Sample File Upload Form
5	DBDEMO_CALENDAR	Calendar Lookup Form
6	DBDEMO_CAPTCHA	Captcha Demo Form

The example above is on a form with fields as shown below. Note that the *REPORT* field is the fifth field on the form.



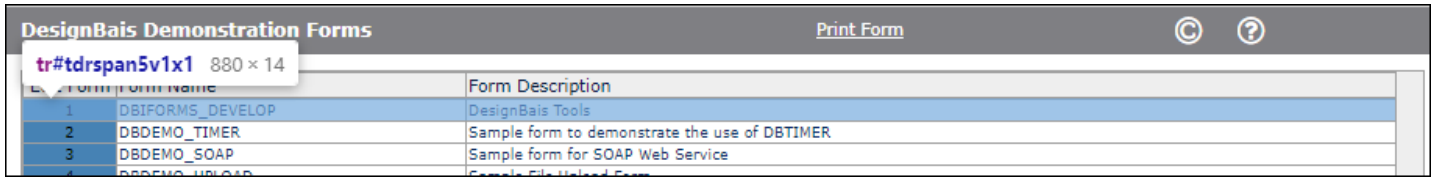
Field Property	Field Name	Att	Mv	Process After
TEXT	DBDESIGNERTEXT			
BUTTON	B.DBICOMMON			DBI.G.BUTTON
BUTTON	B.DBIFORMHELP			DBI.G.BUTTON
BUTTON	B.DBIPRINTFORM			DBI.G.PRINTFORM
REPORT	R.DEMO			DBI.I.DEMO

Inspecting the form in Chrome displays the structure (hover the mouse over a cell of the report, right-click and select *Inspect*)



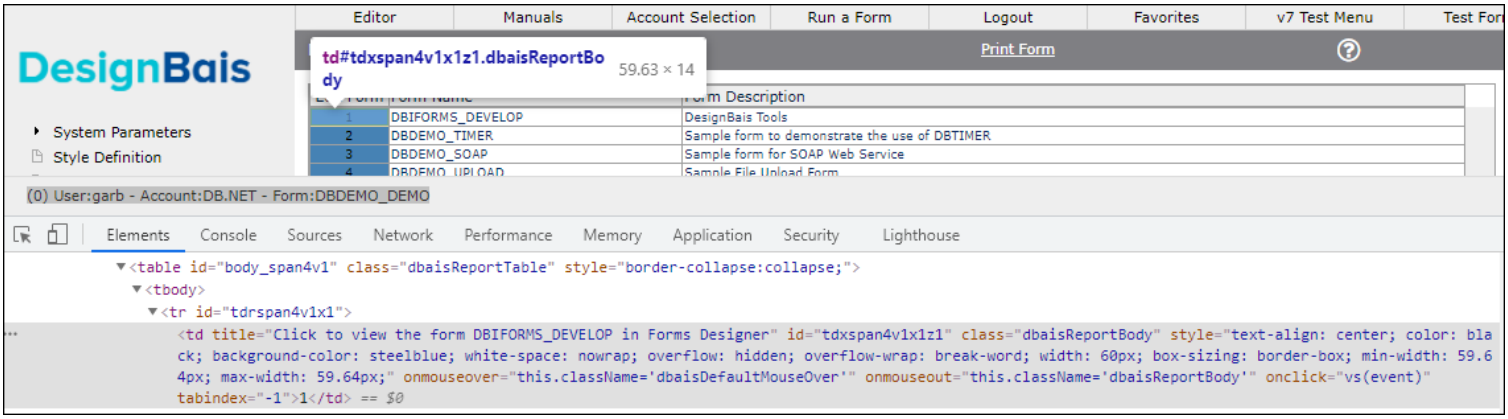
```
▼<div id="span5v1" class="dbaisReportBorder" style="position: absolute; z-index: 1; top: 60px; left: 233px; width: 900px; height: 742px; displ
ay: block; overflow: hidden; box-sizing: border-box;"> == $0
  ▼<div id="divX_span5v1" style="height: auto; width: 898px; overflow: hidden; max-width: 900px;">
    ▶<table id="header_span5v1" style="border-collapse:collapse;">…</table>
    </div>
  ▼<div id="divY_span5v1" class="dbaisReportDetail" onscroll="synchronizeScroll('divY_span5v1','divX_span5v1')" style="margin: 0px; padding: 0
  px; box-sizing: border-box; height: 725px; width: 900px; overflow: auto;">
    ▼<table id="body_span5v1" class="dbaisReportTable" style="border-collapse:collapse;">
      ▼<tbody>
        ▼<tr id="tdrspan5v1x1">
            <td title="Click to view the form DBIFORMS_DEVELOP in Forms Designer" id="tdxspan5v1x1z1" class="dbaisReportBody" style="text-align:
            center; color: black; background-color: steelblue; white-space: nowrap; overflow: hidden; overflow-wrap: break-word; width: 60px; bo
            x-sizing: border-box; min-width: 60px; max-width: 60px;" onmouseover="this.className='dbaisDefaultMouseOver'" onmouseout="this.class
            Name='dbaisReportBody'" onclick="vs(event)" tabindex="-1">1</td>
            <td title="Click to run form DBIFORMS_DEVELOP" id="tdxspan5v1x1z2" class="dbaisReportBody" style="text-align: left; background-colo
            r: white; white-space: nowrap; overflow: hidden; overflow-wrap: break-word; width: 224px; box-sizing: border-box; min-width: 224px;
            max-width: 224px;" onmouseover="this.className='dbaisDefaultMouseOver'" onmouseout="this.className='dbaisReportBody'" onclick="vs(ev
            ent)" tabindex="-1">DBIFORMS_DEVELOP</td>
            <td id="tdxspan5v1x1z3" class="dbaisReportBody" style="white-space: nowrap; overflow: hidden; overflow-wrap: break-word; width: 596p
```

Note that the top row of the report is a *tdrspan* element. The *5* indicates that it is part of the 5th field on the form, the *REPORT* field called *R.DEMO*.. The *v1* indicates the value in DBWLEVEL where 1 means this is a base form.. The *x1* indicates the row number.



Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Hover on the first *<td title=…….</td>* element as shown in the above snip:



This displays the title of the <td> element which is a cell of a table containing data:

td#tdxspan4v1x1z1

- 4 – this is the fourth field on the DesignBais form
- v1 – this is the base form, v2 indicates a modal form 1 layer up from the base form (from DBWLEVEL)
- x1 – row 1 of the on-form report
- z1 – column 1 of the on-form report

A browser running a DesignBais form will display an *undefined* error like that shown below if the basic subroutine called from the form makes reference to an on-form report cell that is not currently rendered in the browser. Note that this is in contrast to the behaviour of DesignBais V6 running on IE where such errors were ignored by the browser.

In the example above the "*tdxspan223v1xz8*" indicates:

- 223 – the error relates to the REPORT field number 223
- v1 – this is a base form
- x – the row is undefined – this is the cause of the error
- z8 – column 8 of the on-form report

The probable cause of the error is that the basic subroutine is using DBREPORT.UPDATE to set the background color of a cell and the row is not defined. It may be the case that the undefined errors are because the report does not exist in the next level and DBREPORT.UPDATE changes must be applied in the MODAL RETURN event rather than in the REPORT event.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Subroutine example – On-form Report

```
SALES.REPORT:
   REPORT.NUMBER=1                          ;*      Refers to the report number on the form
   * Set report variables to null
   OUTPUT.REPORT (REPORT.NUMBER) = ""       ;*      Store output in a report format
   OUTPUT.KEYS(REPORT.NUMBER) = ""          ;*      Store key-links to OUTPUT.REPORT
   OUTPUT.HEADERS(REPORT.NUMBER)= ""        ;*      Store report headers
   OUTPUT.ATTR(REPORT.NUMBER) = ""          ;*      Control report attributes
   OUTPUT.WIDTHS(REPORT.NUMBER) = ""        ;*      Variable to store widths
   OUTPUT.ANCHOR(REPORT.NUMBER) = 1        ;*       Column one will have a click event associated
   *                                                        with it. (The grey column above)
   * Set up the Headers
   *
   OUTPUT.HEADERS(REPORT.NUMBER)<-1> = "Sale Date"
   OUTPUT.HEADERS(REPORT.NUMBER)<-1> = "Product"
   OUTPUT.HEADERS(REPORT.NUMBER)<-1> = "Quantity"
   OUTPUT.HEADERS(REPORT.NUMBER)<-1> = "Unit Price"
   OUTPUT.HEADERS(REPORT.NUMBER)<-1> = "Extended Price"
   OUTPUT.HEADERS(REPORT.NUMBER)<-1> = "Tax"
   OUTPUT.HEADERS(REPORT.NUMBER)<-1> = "Total (Inc Tax)"
   * Set up the widths for each column
   OUTPUT.WIDTHS(REPORT.NUMBER)<-1> = "8"   ;*      Controls the width of each column
   OUTPUT.WIDTHS(REPORT.NUMBER)<-1> = "20"
   OUTPUT.WIDTHS(REPORT.NUMBER)<-1> = "7"
   OUTPUT.WIDTHS(REPORT.NUMBER)<-1> = "8"
   OUTPUT.WIDTHS(REPORT.NUMBER)<-1> = "11"
   OUTPUT.WIDTHS(REPORT.NUMBER)<-1> = "10"
   OUTPUT.WIDTHS(REPORT.NUMBER)<-1> = "11"
   *
   OUTPUT.AT = ''                           ;* Build the alignment requirement for each column
   * Establish column alignments – Default is alignleft; alternatives are aligncenter and alignright
   OUTPUT.AT<1,1> = 'alignright'
   OUTPUT.AT<1,3> = 'alignright'
   OUTPUT.AT<1,4> = 'alignright'
   OUTPUT.AT<1,5> = 'alignright'
   OUTPUT.AT<1,6> = 'alignright'
   OUTPUT.AT<1,7> = 'alignright'
   * Main report section
   TOTQTY=0
   TOTPRICE=0
   TOTTAX=0
   TOTTOTAL=0
   *
   NUM.LINES = DCOUNT(DBOTHER.RECORD(1)<DBS.DATE>,VM)
   FOR DL = 1 TO NUM.LINES
      TOTQTY += DBOTHER.RECORD(1)<DBS.QTY,DL>
      TOTPRICE += DBOTHER.RECORD(1)<DBS.EXTENDED.PRICE,DL>
      TOTTAX += DBOTHER.RECORD(1)<DBS.TAX,DL>
      TOTTOTAL += DBOTHER.RECORD(1)<DBS.TOTAL,DL>
      * Set-up an output line on the report - each value of the line represents a column on the report
      OUTPUT.LINE = OCONV(DBOTHER.RECORD(1)<DBS.DATE,DL>,'D4/')
      OUTPUT.LINE<1,2> = DBOTHER.RECORD(1)<DBS.PRODUCT,DL>
      OUTPUT.LINE<1,3> = OCONV(DBOTHER.RECORD(1)<DBS.QTY,DL>,'MD0,')
      OUTPUT.LINE<1,4> = OCONV(DBOTHER.RECORD(1)<DBS.PRICE,DL>,'MD2,')
      OUTPUT.LINE<1,5> = OCONV(DBOTHER.RECORD(1)<DBS.EXTENDED.PRICE,DL>,'MD2,')
      OUTPUT.LINE<1,6> = OCONV(DBOTHER.RECORD(1)<DBS.TAX,DL>,'MD2,')
      OUTPUT.LINE<1,7> = OCONV(DBOTHER.RECORD(1)<DBS.TOTAL,DL>,'MD2,')
      * Add each line to the OUTPUT.REPORT.VARIABLE
      OUTPUT.REPORT(REPORT.NUMBER)<-1> = OUTPUT.LINE
      * Add the attributes to the OUTPUT.ATTR variable
      OUTPUT.ATTR(REPORT.NUMBER)<-1> = OUTPUT.AT
      * In order for the click event to have a key behind it we add the key to OUTPUT.KEYS
      OUTPUT.KEYS(REPORT.NUMBER)<-1> = DBOTHER.RECORD(1)<DBS.INVOICE.KEYS,DL>
   NEXT DL
   * Add a total line to our report
   OUTPUT.LINE = ''
   * Set-up the descriptions and totals in the variable OUTPUT.LINE
   IF NUM.LINES > 1 THEN
      OUTPUT.LINE<1,2> = 'Total'
      OUTPUT.LINE<1,3> = OCONV(TOTQTY,'MD0Z,')
```

```
   OUTPUT.LINE<1,4> = ''
   OUTPUT.LINE<1,5> = OCONV(TOTPRICE,'MD2Z,')
   OUTPUT.LINE<1,6> = OCONV(TOTTAX,'MD2Z,')
   OUTPUT.LINE<1,7> = OCONV(TOTTOTAL,'MD2Z,')
END ELSE
   OUTPUT.LINE<1,2> = 'No Sales Transactions'
   OUTPUT.LINE<1,1> = 'Client Name'
   OUTPUT.LINE<1,4> = 'Line 4'
END
* Add the total line to the report
OUTPUT.REPORT(REPORT.NUMBER)<-1>=OUTPUT.LINE
OUTPUT.ATTR(REPORT.NUMBER)<-1>=OUTPUT.AT
* Set the refresh parameters
PROCESS.TYPE='R'
PROCESS.REFRESH='R.SALES'                       ;* Refresh the report with the updated details
PROCESS.REFRESH<2>=40                           ;* set OFR page limit to 40 rows for this OFR
```

OUTPUT.REPORT*(n)*

This variable contains the detail lines of each report. This is a subscripted array of ten usable elements that correspond to the report controls loaded onto a form.

Each line on the report (except headers) is reflected as an attribute in this variable

Each cell/column on a line is reflected as a value position.

OUTPUT.HEADERS*(n)*

Is used to define the header row at the top of the On-form Report. This is a subscripted array of ten usable elements that correspond to the report controls loaded onto a form.

Each column of the header is represented as an attribute in the OUTPUT.HEADERS*(n)* record

OUTPUT.WIDTHS*(n)*

Is used to define the width of each column on the report. This is a subscripted array of ten usable elements that correspond to the report controls loaded onto a form. Note that these widths are not absolute width in pixels. They define the proportion of the total width to be allocated to each field. Widths of 20, 20, 40 for example would allocate half the available width to the 3rd column.

In order to define absolute widths in pixels set the width of the first column as “A” concatenated with the required width in pixels, such as “A120”. The width of all other columns will then be treated as absolute pixel widths. The value concatenated to the “A” must be numeric.

Each column width is represented as an attribute in the OUTPUT.WIDTHS*(n)* record.

OUTPUT.ATTR*(n)*

Is used to define the non-standard display attributes of each cell on the report. This is a subscripted array of ten usable elements that correspond to the report controls loaded onto a form.

Each cell on the report is represented as an attribute (row) and value (column) position in the OUTPUT.WIDTHS*(n)* record.

Typical usage of OUTPUT.ATTR

alignright	Right justify the data in the cell
alignleft	Left justify the data in the cell. This is the default setting
aligncenter	Center Justify the data in the cell.

Multiple display attributes/properties can be used in the one setting. Each property must be separated by a “~|” combination.

Example:
Make the color of the data in a particular cell red if the number to print is less than zero:

```
IF NUMBER.TO.PRINT >= 0 THEN
   OUTPUT.ATTR(REPORT.NUMBER)<ROW,COL> = “alignright”
END ELSE
   OUTPUT.ATTR(REPORT.NUMBER)<ROW,COL> = “alignright~|color:red”
END
```

Make the background color of the right-justified cell yellow, the foreground blue, with padding:

```
OUTPUT.ATTR(REPORT.NUMBER)<ROW.COL> = “alignright~|background-color:yellow~|color:blue~|padding-right:5px”
```

The background-color and color properties are standard HTML style properties. You can use most standard HTML style properties in OUTPUT.ATTR to affect the look of the report.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

OUTPUT.ANCHOR*(n)*

Is used to define which columns are to have click events associated with them. This is a subscripted array of ten usable elements that correspond to the report controls loaded onto a form.To assign a click event to a column it is simply a matter of defining the columns (multivalue delimited)

Usage: OUTPUT.ANCHOR(subscript) = *Column Number[:VM:Column Number]*

A click event is to be assigned to columns one and four.

```
OUTPUT.ANCHOR(REPORT.NUMBER) = 1:VM:4
```

For the click event to be active a corresponding cell in OUTPUT.KEYS must have a value.

In Release 6 OUTPUT.ANCHOR(REPORT.NUMBER)<1> was a multivalued list of columns with a click event. From Release 8.7.0.1 this became mult-valued by REPORT.SUBELEMENT with columns in the mutli-subvalues:

```
OUTPUT.ANCHOR(REPORT.NUMBER)<1,REPORT.SUBELEMENT,click column>
```

OUTPUT.KEYS*(n)*

Is used to define which cells have a key assigned to support a click event. This is a subscripted array of ten usable elements that correspond to the report controls loaded onto a form.

Each line on the report is reflected as an attribute in this variable

Each cell/column on a line is reflected as a value position. When the users clicks an active cell on a report the following events take place:

- The value of the OUTPUT.KEYS*(n)* variable is moved to DBVALUE so it can be referenced in a subroutine.
- The row and column is moved to DBREPORT.CELL as *ROW.COL*
- The PROCESS.EVENTSOURCE is set to the name of the On-form Report.
- The "*REPORT*" PROCESS.EVENT is fired.

Usage: OUTPUT.KEYS(subscript)<row,col> = *Data Value*

Example: OUTPUT.KEYS(REPORT.NUMBER)<ROW,COL> = MYKEY

OUTPUT.MOUSEOVER*(n)*

This variable is used to define the mouseover and mouseout classes for the On-form Report.

These classes must be set up in the style definitions.

Usage:

```
OUTPUT.MOUSEOVER(ReportNumber)<Row,Col> = "defaultClass~|mouseOverClass"
```

The DesignBais mouseover style (dbaisDefaultMouseOver) does not include any colors. The mouseover style will override the display properties of the On-form Report column to which it is applied. This means that the anchor column of a report, which normally is highlighted in blue, will have no color if mouseover has been applied.

If color is required then your preferred colors can be added to your mouseover style.

OUTPUT.TYPE*(n)*

The OUTPUT.TYPE(ReportNumber) variable is used to define whether a field on an On-form Report is designated as input.

Usage:

OUTPUT.TYPE(ReportNumber)<Row,Col> = “I”

I – Input Field

After Release 8.6.0.1 the following additional input types can be specified:

IA – Input Field alpha
ID – Input Field date
IN – Input Field numeric (integer)
INn – Input Field numeric (n decimal places; n must be numeric 1 through 9)

Setting the System Parameter *Show Popup Calendar* to *On Focus* will trigger the calendar display when the On-form report date input cell gains focus.

All fields designated as input automatically invoke a server call when the field is changed. DBVALUE holds the value from the field. Note that setting the report column to to have an associated click event using OUTPUT.ANCHOR will prevent the input behaviour of any cell in the column that is an input type cell.

There is no default common variable for the storage of these values apart from DBVALUE, which is the same as any On-form Report cell. You must provide for the storage and validation of these cells in your basic subroutine. It is recommended that a DBSTORE(n) variable be used for the storage of the input cells. Standard DesignBais routines will sort these columns as a normal On-form Report is sorted. You need to ensure that your storage variables are also sorted, if this function is used.

HTML encoding can be applied to an entire on-form report or to a particular column by setting OUTPUT.TYPE(ReportNumber) <Row,Col> = “Y” or “E”.

For an individual cell you can set OUTPUT.TYPE(N)<ROW,COL> to override the report field setting for HTML Encoding:

- I input field, no encoding applied
- T textarea input
- F or IF or TF add onfocus event as well as the onchange for the input
- MouseOver add mouse over event
- MouseOverOut add mouse over and mouse out event
- Y encode
- E encode, same as Y to retain compatibility with an earlier release
- N no encoding
- Null use field default, system setting or global setting (in that order) for HTML encoding

OUTPUT.TITLE*(n)*

The OUTPUT.TITLE(n) variable is used to declare text to be added as mouse-over titles for each cell on an on-form-report.

Usage:

OUTPUT.TITLE(*Number*)<Line,Cell> = *“The title for the cell”*

Eg.

OUTPUT.TITLE(1)<ROWCOUNT,COLUMNPOS> = “Help text for the cell”

PROCESS.TYPE

Combined with PROCESS.REFRESH, defines the type of form object that needs to be refreshed with On-form Report variables. Valid values are “R” for Report and H for Highchart. The variable consists of a single attribute with multiple values. Each value corresponds to a Process Report Number.

Eg: PROCESS.TYPE = “R” :VM: “H”

The above example designates a Report and a Highchart, the names of which are passed in the associated value position in PROCESS.REFRESH.

From Release 8.3.3.4 PROCESS.TYPE can also have the value “RP” to indicate page processing. This is used in association with PROCESS.REFRESH<2> as explained below.

PROCESS.REFRESH

Combined with PROCESS.TYPE, specifies the specific form object that needs to be refreshed with On-form Report variables. Variable consists 2 attributes with multiple values, each value corresponds to an object on the form.

Eg: PROCESS.REFRESH = “R.SALES” :VM: “G.SALES”
PROCESS.REFRESH<2> = 40:VM:30

This indicates that the form report element is called R.SALES and the Highchart is called G.SALES.
The second attribute sets the number of rows per page for the On-form Report. This is not meaningful for the graph and should be left blank.

Setting PROCESS.REFRESH<2> to the following alpha values in association with PROCESS.TYPE = ‘RP’ allows page processing:

P display previous page
N display next page
F display first page
L display last page

Use PROCESS.REFRESH<3,nn> to specify the number of fixed columns in an On-form Report. One or more columns, commencing with the first column can be fixed, or sticky, meaning that they remain in view when using the horizontal scroll bar. Fixed columns must be contiguous.

Use PROCESS.REFRESH<4,nn> to specify the starting page number.

DBPAGEDEPTH

If rows are collapsed in an OFR then child rows appearing on subsequent pages also need to be collapsed. In order to achieve this the OFR page breaks will not necessarily occur at a constant row count increment.

DBPAGEDEPTH<PROCESS.REPORT.NUMBER,REPORT.SUBELEMENT,PGNO> is structured like OUTPUT.ANCHOR. DesignBais will force the REPORT.SUBELEMENT if not supplied by the developer. In the following example the developer calculates the required page depth and stores it in DBPAGEDEPTH. It is then used to set the on-form report page size value in PROCESS.REFRESH<2>:

DBPAGEDEPTH<PROCESS.REPORT.NUMBER,SUBREPNO,PGNO> = PG.RCNT
PROCESS.REFRESH<2,-1>=DBPAGEDEPTH<PROCESS.REPORT.NUMBER,SUBREPNO,1>

DBPAGEDEPTH functions like OUTPUT.ANCHOR in that the developer can use it as a MV list by page unless there are sub-reports or the developer can use SVM even when there is no sub-element report.

OUTPUT.ANCHOR(REPORT.NUMBER)<1,REPORT.SUBELEMENT,click column>

DBREPORTROW.ATTR

DBREPORTROW.ATTR<ROW> is used to define attributes for an entire row of an on-form report, for example, show or hide a row. The rows are controlled by OUTPUT.REPNUMBER(REPORT.NUMBER) in the same way as OUTPUT.REPORT.

The Class for a row may be set with *class=fred.* Note that quotes will be removed and re-added by the DesignBais.

Your own attributes may also be added via *attrName=fred* for use with your own javascript.
For example:

```
DBREPORTROW.ATTR<PROCESS.REPORT.NUMBER,RCNT> = "dblevel=":LVL:"~|display:none"
```

Note that OUTPUT.ATTR defines attributes for individual cells of the report rather than entire rows.
Developers should review the Demo form DBDEMO*OFR.HIDEROWS available from the top menu of the DesignBais Tools form. This form utilises these common variables in order to hide and display the exploded lines of a bill of materials.

DBREPORTROWCLASS

Allows the developer to set onmouseover and onmouseout classes for all rows of an On-Form Report.

Any OFR defaults will be overwritten by this setting.

Any OUTPUT.MOUSEOVER attributes will still be applied to report cells.

```
DBREPORTROWCLASS<REPORT.SUBSCRIPT,1,SUB.ELM> = “MouseOverClass”
DBREPORTROWCLASS<REPORT.SUBSCRIPT,2,SUB.ELM> = “MouseOutClass”
```

If no report sub-element is defined then SUB.ELM = 1 so that the assignment can be of the form:

```
DBREPORTROWCLASS<REPORT.SUBSCRIPT,1> = “MouseOverClass”
DBREPORTROWCLASS<REPORT.SUBSCRIPT,2> = “MouseOutClass”
```

On-form Reports – Clearing the Report

To clear data from the report container it is necessary to refresh the report with no data.

If you want an empty report with column headings then send the header details.

```
PROCESS.REPORT.NUMBER = n
OUTPUT.REPORT(PROCESS.REPORT.NUMBER) = ""
OUTPUT.ATTR(PROCESS.REPORT.NUMBER)   = ""
OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER) = 1  ;* one column with no header or data
PROCESS.TYPE='R'
PROCESS.REFRESH='R.reportname'
```

Controlling Heading Attributes On-form Reports

DBREPORTHEADER.ATTR

As a form is rendered column headings for On-form Reports can be controlled by the common variable DBREPORTHEADER.ATTR.

Each attribute of this variable corresponds to the number of a report.
Each value corresponds to a column in the report.
You can add a sub-element to the report number. So report number 1 can become 1.1, 1.2, 1.3 etc. In this case specify the sub-element number in the subvalue position of DBREPORTHEADER.ATTR.

DBREPORTHEADER.ATTR<repNo,colNo,subRep>

Attributes are separated by "~|"

Eg:

REPORT.NUMBER = 1
DBREPORTHEADER.ATTR<REPORT.NUMBER> = "background-color:red~|color:blue~|padding-right:5px"

This will set foreground and background colors for the header of the first column of the first report on a form. Note the use of css format not javascript format.

Inserting Break Row inText in On-form Reports

Use the string '@BR@' in text to force a break row in the on-form report display. This string is converted to the HTML break row.

Example:

```
* FULL.DESC is a multivalue text field to be appended to the title and separated by a break row
DELIM = '@BR@'
FOR LL=1 TO LMAX
   CHK.PAGE.TITLE := DELIM:FULL.DESC<1,LL>
NEXT LL
```

Playing a Sound in On-form Reports

This example shows how to generate a sound when an on-form report cell is clicked.

Add a hidden field to your form and set the field value in the After Display event. For example:

```
* Add audio
DBWORK<DEM.WORK1.WK> = '<audio id="ding" autoplay="" src="images\dbinetdemo\ding-sound-effect.mp3">'
```

Assume that column 5 of the on-form report is to play a sound when any row is clicked:

```
OUTPUT.ANCHOR(REPORT.NUMBER)  = "5" ;* Column will have a click event associated with it.
OUTPUT.HEADERS(REPORT.NUMBER)<5> = "Sound"
OUTPUT.LINE<1,5> = ' '
OUTPUT.KEYS(REPORT.NUMBER)<DL,5> = DL
```

In the REPORT event, when a cell is clicked:

```
DBAJAXCMD<-1> = 'document.getElementById("ding").play();   '
```

On-form Reports – Changing the values of cells after a report is displayed

There may be times where it is necessary to change the value or display attribute/property of a cell within an On-form Report.

DBREPORT.UPDATE
DBREPORT.CELL

These variables provides the developer the ability to change the characteristics of a cell at runtime. This is useful in situations where the cell is to indicate that the user has selected a particular cell, or as a result of another action a value in a cell has to be changed. This removes the requirement of redisplaying the entire report.

Usage:

```
DBREPORT.CELL           Holds the row and column of the clicked cell in the form Row.Col

DBREPORT.UPDATE<1,n>    Name of the report to update
DBREPORT.UPDATE<2,n>    Row of the cell to be changed.
DBREPORT.UPDATE<3,n>    Column of the cell to be changed
DBREPORT.UPDATE<4,n>    Data value [If blank there is no change to the data in the cell]
DBREPORT.UPDATE<5,n>    Display Attributes. These are not to be stacked as per OUTPUT.ATTR
DBREPORT.UPDATE<6,n>    Input flag. If updating an input type cell set this flag to "I"
DBREPORT.UPDATE<7,n>    Null toggle. Set to 1 to set the value of the cell to null
DBREPORT.UPDATE<8,n>    The title of the cell. Amend the OUTPUT.TITLE value for the cell.
DBREPORT.UPDATE<9,n>    The key assigned to the cell. Amend the OUTPUT.KEYS value for the cell.
DBREPORT.UPDATE<10,n>   Flag "P" to apply styling to the parent TD, or set to the child number
```

If more than one attribute is to be changed, a unique value is required for each of the DBREPORT.UPDATE fields.

In Display Attributes use javascript style properties only and use "~|" to separate style property and value in the one multivalue.

DesignBais sends the javascript command: document.getElementById("**xmllabel**").style.**property=value**;
For example: document.getElementById("**xmllabel**").style.**backgroundColor**="**yellow**";

DBREPORT.UPDATE<10> overcomes the DesignBais method of assigning an id to the cell (input cell for example) but not to the container (the TD). Setting a value of "P" in attribute 10 causes DesignBais to apply the styling to the parent TD. Refer to the code example in the list of BASIC.SKELETON code samples accessed from the *Subroutine* button on the Code Editor form. In this example the original <td> items do not have a background-color. During the focus event we add a background-color to the <td>'s in the highlighted row.

Alternatively DBREPORT.UPDATE<10> can specify the *child number*. This is an integer that refers to the required element within your styled OFR input field. For example you may have an awesome font within the input cell.

Rather than utilizing the "!important" javascript notation which can be messy DesignBais provides a "REMOVE" function to do a ".style.removeProperty".

```
DBREPORT.UPDATE<5,CU> = 'background-color~|REMOVE'    ;* Removes the style property
```

Note that the removeProperty needs the **HTML name "background-color"** rather than the normal DBREPORT.UPDATE javascript syntax "backgroundColor".

Example:

The user has clicked a cell. The clicked cell needs to change its colors to a yellow background with red data.

```
ROW = FIELD(DBREPORT.CELL,".",1)
COL = FIELD(DBREPORT.CELL,".",2)

DBREPORT.UPDATE = ""
DBREPORT.UPDATE<1,1> = "R.SALES"
DBREPORT.UPDATE<2,1> = ROW
DBREPORT.UPDATE<3,1> = COL
DBREPORT.UPDATE<4,1> = "" ; * No change to data value
DBREPORT.UPDATE<5,1> = "backgroundColor~|yellow"

DBREPORT.UPDATE<1,2> = "R.SALES"
DBREPORT.UPDATE<2,2> = ROW
DBREPORT.UPDATE<3,2> = COL
DBREPORT.UPDATE<4,2> = "" ; * No change to data value
DBREPORT.UPDATE<5,2> = "color~|red"
```

Only javascript style properties can be used in DBREPORT.UPDATE and "~|" must be used to separate the style property from the value assigned to the property. For example "backgroundColor~|yellow".

DBREPORT.UPDATE triggers javascipt code to update HTML elements within the On-form Report. The display elements in attribute 5 must comply with javascript naming conventions which are case sensitive.

This is why it is necessary to use "backgroundColor" within DBREPORT.UPDATE<5> whereas the HTML loaded into OUTPUT.ATTR when building the On-form Report would require, in this example, "background-color". Javascript would wrongly interpret the hyphen in "background-color" as a subtraction operation.

If your application is using PROCESS.STACK from within the REPORT event then note that DBREPORT.UPDATE is applied after the DBWLEVEL has been incremented. This means that changes to a report cell in the current form must be applied in the MODAL RETURN event as you return from the the stacked form. Use DBSTORE to retain the row and column values and reset these on return. (If you observe form element undefined errors it is because the report does not exist in the next level and DBREPORT.UPDATE changes must be applied in the MODAL RETURN.)

If a process stack call is generated by the click on the report cell then position the DBREPORT.UPDATE in the modal return from the stacked form. You will have to store the row and column values, using DBSTORE for example, in the REPORT event for use in the MODAL RETURN event.

In Release 7 and above developers must ensure that the on-form report cell that is referenced by DBREPORT.UPDATE is currently rendered on the form. If it is not then the browser may display an "undefined" error message. In practice it is often the case that the on-form report that is being referenced is either hidden, or is part of another form that is in the level below the current form.

On-form Reports – Scrolling to a Specified Row

DesignBais provides the ability to scroll to a selected row in the displayed page of an On-form Report.

DBSCROLLREPORT

This variable comprises three multi-valued attributes.

Usage:

DBSCROLLREPORT <1,n>	Name of the report in which to adjust scroll depth.
DBSCROLLREPORT <2,n>	The depth (in pixels) from the top row of the report to the top of the scrollbar.
DBSCROLLREPORT <3,n>	The space (in pixels) from the left margin of the report to the scrollbar.

After the On-form Report has been displayed use a separate event, such as a button click for example, to set DBSCROLLREPORT.

Use DBSCROLLREPORT <2,n> = “scrollHeight” (case insensitive) if you want to scroll to the bottom of the report.
Use DBSCROLLREPORT <3,n> = “scrollWidth” (case insensitive) if you want to scroll all the way to the right of the report.

The following is an example of the code that could be used to scroll an on-form report:

```
POS = DCOUNT(DBSCROLLREPORT<1>,VM)+1
BEGIN CASE
   CASE EVENTSOURCE = "B.START"
      DBSCROLLREPORT<1,POS> = 'R.REPORT1'
      DBSCROLLREPORT<3,POS> = '0'
   CASE EVENTSOURCE = "B.END"
      DBSCROLLREPORT<1,POS> = 'R.REPORT1'
      DBSCROLLREPORT<3,POS> = 'SCROLLWIDTH'
   CASE EVENTSOURCE = "B.TOP"
      DBSCROLLREPORT<1,POS> = 'R.REPORT1'
      DBSCROLLREPORT<2,POS> = '0'
   CASE EVENTSOURCE = "B.BOTTOM"
      DBSCROLLREPORT<1,POS> = 'R.REPORT1'
      DBSCROLLREPORT<2,POS> = 'SCROLLHEIGHT'
```

In general the calculation of the depth in pixels will be:

SCROLL.DEPTH = (NO.OF.ROWS * ROW.DEPTH) – ROW.DEPTH

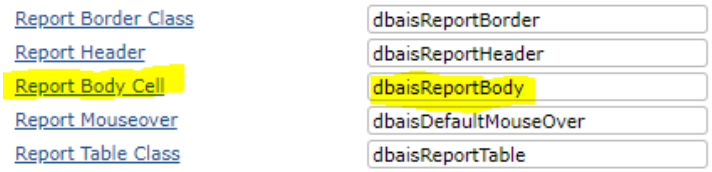
Example:

If the requirement is to scroll to the 10th row of a report called R.REPORT1 where the row depth of the report is 14 pixels then:

SCROLL.DEPTH = (10 * 14) – 14
SCROLLDEPTH = 126
DBSCROLLREPORT <1> = R.REPORT1.
DBSCROLLREPORT <2> = 126

In order to implement this feature it is necessary to know the row height of the On-form Report. For a particular report this can be found by using the browser *Inspect* feature (right click on the report). Alternatively a *Height* property can be

added to the *Report Body Cell* style within the *dbaisWeb* style group. This standard row height can then be used in the scroll depth calculation.



If paging has been applied to the On-form Report, either by setting attribute 2 of the common variable PROCESS.REFRESH or via the System Parameters *Max Rows per OFR Page* setting, then the value in DBSCROLLREPORT is applied to the currently displayed page of the On-form Report.

In the example below DBSCROLLREPORT is used to re-display from a designated row an on-form report that uses page size. Note the PROCESS.TYPE = 'RP' is used to re-display the page containing the row to which the report is to scroll. This example is based on the OFR row height is 14 pixels.

```
CASE SCREEN.NO = "Z120" AND EVENTSOURCE='R.REPORT2'
   BEGIN CASE
      CASE COL = 1 OR COL = 2
         DISP.ROW = FIELD(DBVALUE,'|',2)
         * Check for Page Size or Use Known Page Size
         IF PGSIZE = '' THEN
            READV PGSIZE FROM F.DBIPARMS,"0",DBIPM.MAX.OFR.PAGE ELSE PGSIZE = ''
         END
         IF PGSIZE#'' THEN
            PG = INT(DISP.ROW/PGSIZE)
            DISP.ROW = MOD(DISP.ROW,PGSIZE)
            IF DISP.ROW > 0 THEN PG += 1
            * Go to correct page
            PROCESS.REFRESH = 'R.REPORT1'
            PROCESS.REFRESH<2> = PG
            PROCESS.TYPE = 'RP'
         END
         DBSCROLLREPORT = "R.REPORT1"
         DBSCROLLREPORT<2> = (14 * DISP.ROW) - 14
         IF DBSCROLLREPORT<2> < 1 THEN DBSCROLLREPORT<2> = 1
   END CASE
```

On-form Reports – Tabbing through OFR Input Fields

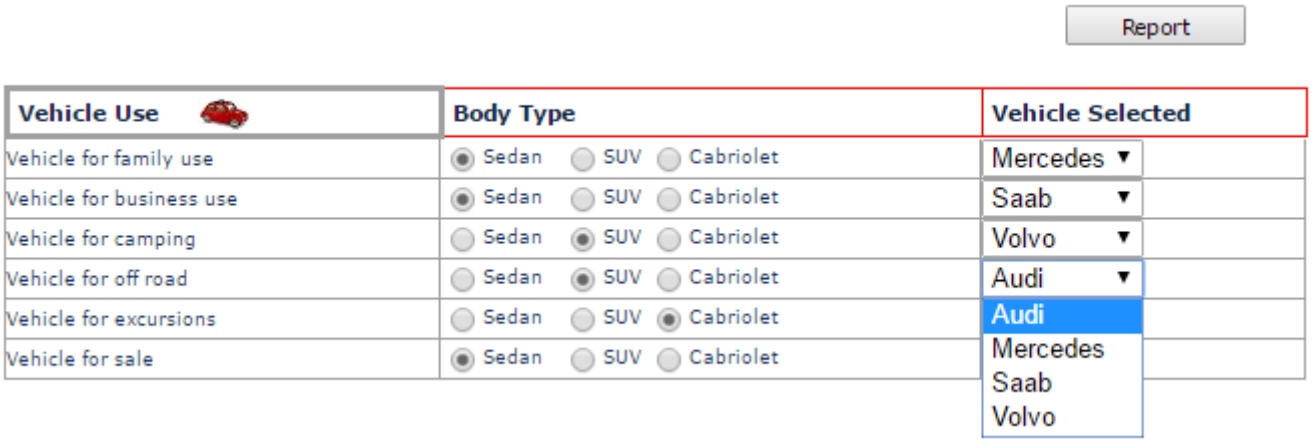
In order to allow tabbing through on-form report input fields it is necessary to check the *Include Reports* option on the front Forms Designer screen. This results in all OFR input fields being added to the Tab Index Properties maintenance form. Any Input fields in the OFR will then be assigned a Tab Index. For backwards compatibility this is optional.

On-form Reports – Adding features to a report

The following example demonstrates how to add features to your On-form Report:

- Formatted Column Headers
- Dropdown Lists
- Radio Buttons

The code to produce this On-form Report is shown below. It is driven from the button “Report” called B.REPORT.




```
*
BUTTON:
* Process to perform after a button is pressed
* EVENTSOURCE = Button Name
      BEGIN CASE

         CASE EVENTSOURCE = "B.REPORT" AND SCREEN.NO = "OFRDROPDOWN"
            GOSUB SETUP.RADIOBUTTON
            GOSUB SETUP.DROPDOWN
            GOSUB BUILD.REPORT
            GOSUB REPORT.REFRESH
      END CASE
      RETURN
*
BUILD.REPORT:
*
      REPORT.NUMBER=1                    ;* Refers to the report number on the form
      *
      OUTPUT.REPORT (REPORT.NUMBER) = "" ;* Store output in a report format
      OUTPUT.KEYS(REPORT.NUMBER)    = "" ;* Store key-links to OUTPUT.REPORT
      OUTPUT.HEADERS(REPORT.NUMBER) = "" ;* Store report headers
      OUTPUT.ATTR(REPORT.NUMBER)    = "" ;* Control report attributes
      OUTPUT.WIDTHS(REPORT.NUMBER)  = "" ;* Variable to store widths
      OUTPUT.ANCHOR(REPORT.NUMBER)  = "" ;* Column one will have a click event associated with it.
      *
      OUTPUT.HEADERS(REPORT.NUMBER)<-1> = "Vehicle Use"
      OUTPUT.HEADERS(REPORT.NUMBER)<-1> = "Body Type"
      OUTPUT.HEADERS(REPORT.NUMBER)<-1> = "Vehicle Selected"
      *
      DBREPORTHEADER.ATTR<REPORT.NUMBER,1> = "border-width:3px;border-style:solid~|background:url(images/smallRedCar.jpg)
no-repeat center~|font-weight:bold"
      FOR DL = 2 TO 3
         DBREPORTHEADER.ATTR<REPORT.NUMBER,DL> = "border-style:solid~|background-color:white~|border-width:1px~|border-
color:red~|font-weight:bold"
      NEXT DL
      *
      OUTPUT.WIDTHS(REPORT.NUMBER)<-1> = "20"
      OUTPUT.WIDTHS(REPORT.NUMBER)<-1> = "25"
      OUTPUT.WIDTHS(REPORT.NUMBER)<-1> = "15"
      *
```

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

```
      OUTPUT.AT = ''
      *
      DESCS = ""
      DESCS<-1> = "Vehicle for family use"   ; VALS<-1> = ""
      DESCS<-1> = "Vehicle for business use" ; VALS<-1> = ""
      DESCS<-1> = "Vehicle for camping"      ; VALS<-1> = ""
      DESCS<-1> = "Vehicle for off road"     ; VALS<-1> = ""
      DESCS<-1> = "Vehicle for excursions"   ; VALS<-1> = ""
      DESCS<-1> = "Vehicle for sale"         ; VALS<-1> = ""
      *
      NUM.LINES=DCOUNT(DESCS,AM)
      FOR DL = 1 TO NUM.LINES
         OUTPUT.LINE      = DESCS<DL>
         OUTPUT.LINE<1,2> = CHANGE(RADIO.HTML,'%R.ROW%',DL)
         OUTPUT.LINE<1,3> = CHANGE(DROPDOWN,'%R.ROW%',DL)
         * Add each line to the OUTPUT.REPORT.VARIABLE
         OUTPUT.REPORT(REPORT.NUMBER)<-1> = OUTPUT.LINE
         OUTPUT.TYPE(REPORT.NUMBER)<-1,2> = "N"
         OUTPUT.TYPE(REPORT.NUMBER)<-1,3> = "N"
      NEXT DL
      RETURN
*
REPORT.REFRESH:
*
      PROCESS.TYPE='R':VM:'R'
      PROCESS.REFRESH='R.REPORT1':VM:'R.FLASH'
      RETURN
*
SETUP.DROPDOWN:
*
      REPORT.FIELD = "R.REPORT1"
      R.ROW = "%R.ROW%"
      R.COL = 3
      DD.LIST = ""
      DD.LIST<-1> = "audi":VM:"Audi"
      DD.LIST<-1> = "mercedes":VM:"Mercedes"
      DD.LIST<-1> = "saab":VM:"Saab"
      DD.LIST<-1> = "volvo":VM:"Volvo"
      DDMAX = DCOUNT(DD.LIST,AM)
      *
      LOCATE REPORT.FIELD IN TABLEDATA(DBIF.FIELD.NAME.LIST)<1> SETTING R.POS ELSE RETURN
      REPORT.XML.LABEL = TABLEDATA(DBIF.FIELD.XML.LABEL)<1,R.POS>
      *
      CHG = ' onchange="validateAndSubmit(event);" '           ;* pre Rel 7
      CHG = ' onchange="vs(event);" '                          ;* Rel 7
      CHG:= 'style="cursor:pointer" '
      *
      DROPDOWN = '<select id="tdxspan':R.POS:'v':DBWLEVEL:'x':R.ROW:'z':R.COL:'" ':CHG:'>'
      FOR DCNT = 1 TO DDMAX
         DROPDOWN := '<option value="':DD.LIST<DCNT,1>:'">':DD.LIST<DCNT,2>:'</option>'
      NEXT DCNT
      DROPDOWN := '<select>'
      RETURN
*
SETUP.RADIOBUTTON:
*
      REPORT.FIELD = "R.REPORT1"
      R.ROW = "%R.ROW%"
      R.COL = 2
      RB.LIST = ""
      RB.LIST<-1> = "S":VM:"Sedan  ":VM:"%SCHECK%"
      RB.LIST<-1> = "U":VM:"SUV":VM:"%UCHECK%"
      RB.LIST<-1> = "C":VM:"Cabriolet":VM:"%CCHECK%"
      RBMAX = DCOUNT(RB.LIST,AM)
      *
      LOCATE REPORT.FIELD IN TABLEDATA(DBIF.FIELD.NAME.LIST)<1> SETTING R.POS ELSE RETURN
      REPORT.XML.LABEL = TABLEDATA(DBIF.FIELD.XML.LABEL)<1,R.POS>
      *
      CLICK = ' onclick="validateAndSubmit(event)" '           ;* pre Rel 7
      CLICK = ' onclick="vs(event)" '                          ;* Rel 7
      CLICK:= 'style="cursor:pointer" '
```

```
RADIO.CLICK = ' <input type="radio" class="dbaisReportBody" ':CLICK:' '
*
RADIO.HTML = ""
FOR DCNT = 1 TO RBMAX
   RADIO.HTML := RADIO.CLICK
   RADIO.HTML :=' id="tdxspan':R.POS:'v':DBWLEVEL:'x':R.ROW:'z':R.COL
   RADIO.HTML :='" name="status':R.ROW:'" value="':RB.LIST<DCNT,1>:'" ':RB.LIST<DCNT,3>:'>':RB.LIST<DCNT,2>
NEXT DCNT
RETURN
```

Notes:

1. TABLEDATA is a common variable defined in DBI in a record called DBI.SUB.COMMON. You must INCLUDE this common in your basic program. TABLEDATA contains the full list of fields on a form including those from any header and footer forms. The position of the report field will be effected by whether a header form exists since all header form fields wil be inserted before the list of fields on the main form.

2. The image “smallRedCar.jpg” referred to in DBREPORTHEADER.ATTR must exist in your images folder on the web server.

3. In the above code the style=”cursor:pointer” is optional but obviously changes the cursor when the radio button gains focus.

4. Border widths must be the same in the Report Detail & Report Header styles otherwise the columns will not align. Any style attribute that affects cell widths will also disturb the column alignment.

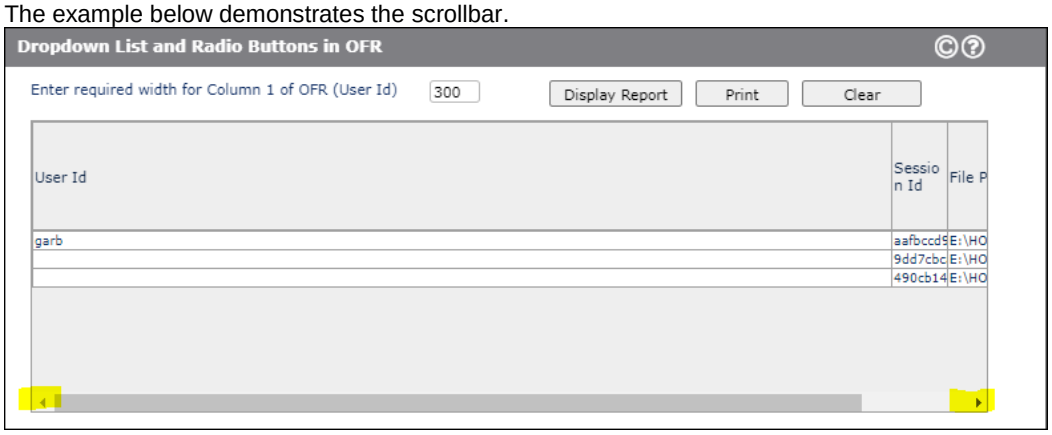
5. The “%SCHECK%”, “%USCHECK%”, “%CCHECK%” strings are required to allow you to set the current value of the radio buttons. Your code will need to read the application database record holding these fields and replace, for example, “%SCHECK%” with the value “checked” if this radio button is checked, or null is this radio button is off. In your REPORT paragraph where you process the results of clicking on a report cell, you will need to update these strings in order to redisplay the updated cell in the On-form Report.

6. PROCESS.TYPE and PROCESS.REFRESH are shown as a multivalued string as an example of how to refresh multiple On-form Reports.

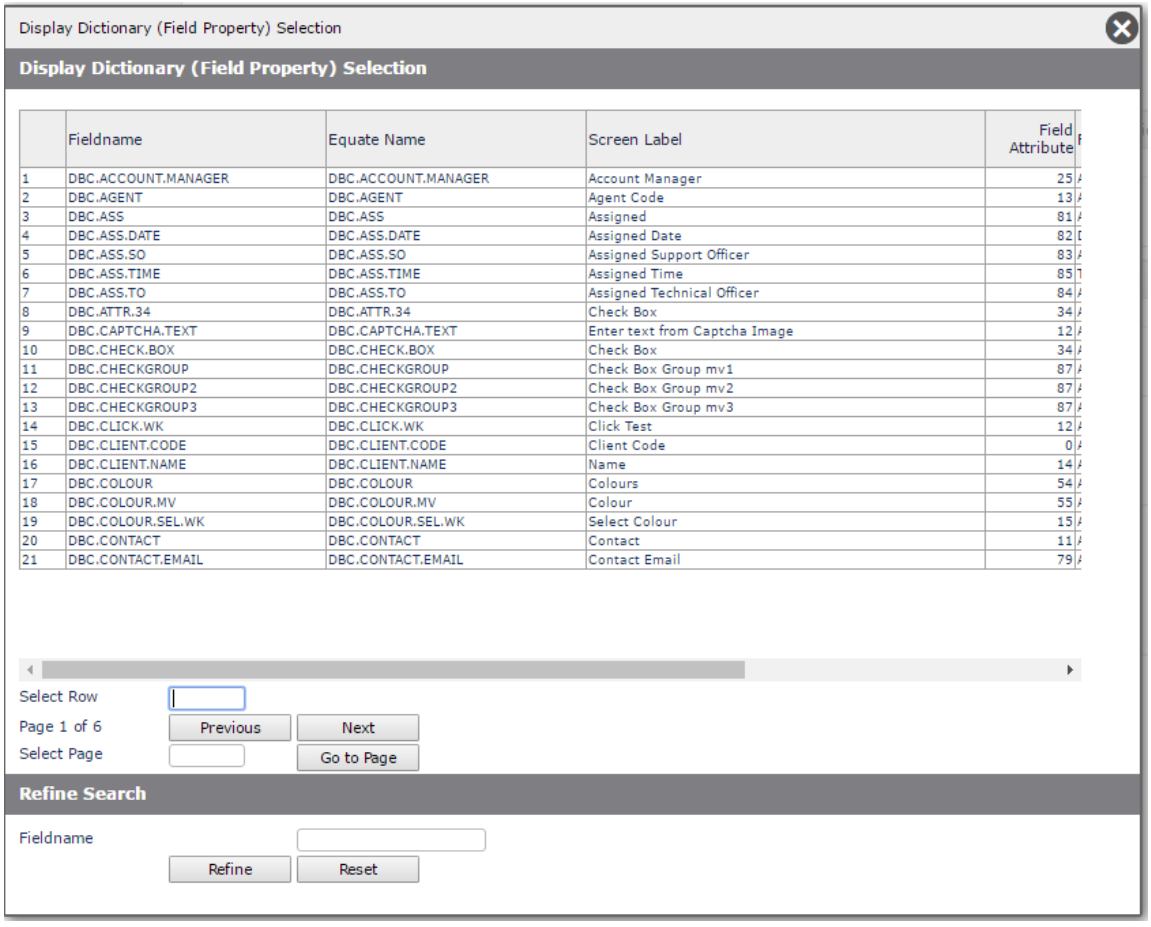
7. The Encode HTML field on the Report Definition field on the form must be set to “N” (or “Inherit” if the System Parameters and General Global Parameters are set to “N”).

8. An On-form Report will display a horizontal scrollbar in the following circumstances:
   - It is on a modal form
   - The Report container column span is expressed as a percentage
   - The width of the report will cause it to extend past the boundary of the form

   The example below demonstrates the scrollbar.



   The Field Property Selection form, from the Selection Process tool, demonstrates the scrollbar:



	Fieldname	Equate Name	Screen Label	Field Attribute
1	DBC.ACCOUNT.MANAGER	DBC.ACCOUNT.MANAGER	Account Manager	25
2	DBC.AGENT	DBC.AGENT	Agent Code	13
3	DBC.ASS	DBC.ASS	Assigned	81
4	DBC.ASS.DATE	DBC.ASS.DATE	Assigned Date	82
5	DBC.ASS.SO	DBC.ASS.SO	Assigned Support Officer	83
6	DBC.ASS.TIME	DBC.ASS.TIME	Assigned Time	85
7	DBC.ASS.TO	DBC.ASS.TO	Assigned Technical Officer	84
8	DBC.ATTR.34	DBC.ATTR.34	Check Box	34
9	DBC.CAPTCHA.TEXT	DBC.CAPTCHA.TEXT	Enter text from Captcha Image	12
10	DBC.CHECK.BOX	DBC.CHECK.BOX	Check Box	34
11	DBC.CHECKGROUP	DBC.CHECKGROUP	Check Box Group mv1	87
12	DBC.CHECKGROUP2	DBC.CHECKGROUP2	Check Box Group mv2	87
13	DBC.CHECKGROUP3	DBC.CHECKGROUP3	Check Box Group mv3	87
14	DBC.CLICK.WK	DBC.CLICK.WK	Click Test	12
15	DBC.CLIENT.CODE	DBC.CLIENT.CODE	Client Code	0
16	DBC.CLIENT.NAME	DBC.CLIENT.NAME	Name	14
17	DBC.COLOUR	DBC.COLOUR	Colours	54
18	DBC.COLOUR.MV	DBC.COLOUR.MV	Colour	55
19	DBC.COLOUR.SEL.WK	DBC.COLOUR.SEL.WK	Select Colour	15
20	DBC.CONTACT	DBC.CONTACT	Contact	11
21	DBC.CONTACT.EMAIL	DBC.CONTACT.EMAIL	Contact Email	79

The following example demonstrates how to merge columns in an On-form Report. This is the equivalent of the merge cells feature of spreadsheets.

Consider this simple report with 5 columns:

Column 1	Column 2	Column 3	Column 4	Column 5
1	Vehicle for family use	4946	-8909	32209
2	Vehicle for business use	9758	-24221	18588
3	Vehicle for camping	6422	-24946	27506
4	Vehicle for off road	3031	-16413	29168
5	Vehicle for excursions	900	-32591	18762
6	Vehicle for sale	1655	-17410	6359

In order to combine the 3 columns highlighted in red simply add the “colspan:n” element to the appropriate column attributes where “n” is the number of colums to span or merge.

```
OUTPUT.AT<1,1>    = "alignleft"
OUTPUT.AT<1,2>    = "aligncenter"
OUTPUT.AT<1,3>    = "alignright"
OUTPUT.AT<1,4>    = 'alignright'
OUTPUT.AT<1,5>    = "aligncenter"
```

becomes

```
OUTPUT.AT<1,1>    = "alignleft"
OUTPUT.AT<1,2>    = "colspan:3~|aligncenter "
OUTPUT.AT<1,3>    = "alignright"
```

```
OUTPUT.AT<1,4>    = 'alignright'
OUTPUT.AT<1,5>    = "aligncenter"
```

Column 1	Column 2	Column 3	Column 4	Column 5
1	Vehicle for family use			21548
2	Vehicle for business use			4041
3	Vehicle for camping			10291
4	Vehicle for off road			11020
5	Vehicle for excursions			27348
6	Vehicle for sale			24484

Notes:

1. The “colspan:n” is case insensitive but must not contain any spaces.
2. The data in the spanned columns is not displayed unless the developer merges the data using the basic subroutine.
3. To combine the headings use DBREPORTHEADER.ATTR<PROCESS.REPORT.NUMBER,2> = "colspan:3~|aligncenter"

Here is an example of code to produce a report using the “colspan:n” feature:

```
NUM.LINES=DCOUNT(NAMES,AM)
   DL = 0
   FOR CNT = 1 TO NUM.LINES
      DL += 1
      OUTPUT.AT<1,1>    = "colspan:3~|alignleft~|font-weight:bold~|background-color:thistle"
      OUTPUT.LINE       = TITLES<CNT>:" ":GIVENS<CNT>:" ":NAMES<CNT>
      OUTPUT.REPORT(REPORT.NUMBER)<-1> = OUTPUT.LINE
      OUTPUT.ATTR(REPORT.NUMBER)<DL> = OUTPUT.AT
      *
      DL += 1
      OUTPUT.AT<1,1>    = ""
      OUTPUT.LINE<1,1> = TITLES<CNT>
      OUTPUT.LINE<1,2> = GIVENS<CNT>
      OUTPUT.LINE<1,3> = NAMES<CNT>
      OUTPUT.REPORT(REPORT.NUMBER)<-1> = OUTPUT.LINE
      OUTPUT.ATTR(REPORT.NUMBER)<DL> = OUTPUT.AT
   NEXT CNT
```

The On-form Report looks like this:

Title	Given Name	Surname
Mr Fred Harry Smith		
Mr	Fred Harry	Smith
Dr Robert John Jones		
Dr	Robert John	Jones
Mrs Fiona Rosemary Brown		
Mrs	Fiona Rosemary	Brown
Ms Stefanie Olive Andrews		
Ms	Stefanie Olive	Andrews
Sir Mark John Cramer		

If you wish to hide a column for a particular report set its width to 0 or indeed skip it altogether. To hide a column for a particular row, use colspan on the previous column.

@PARENT

DesignBais provides an easier method of deriving the element id for use in the HTML required for the on-form report example above.

@PARENT returns the element id and adds ‘xn’ to create a unique identifier for each element in the cell.

So instead of code like this:

```
ID='"tdxspan':R.POS:'v':DBWLEVEL:'x':R.ROW:'z':R.COL:'" '
BTN.HTML = '<input type="button" id=':ID:CHG:' value="OK">'
BTN.HTML := '</input>'
```

you can use the *@PARENT* to return the ID

```
BTN.HTML = '<input type="button" class="BUTTONdbaisDefault" style="height=14px; font-size:10px;"'
BTN.HTML:= ' value="Apply" @PARENT onclick="vs(event);">'
BTN.HTML:= '  <input type="button" class="BUTTONdbaisDefault" style="height=14px; font-size:10px;"'
BTN.HTML:= ' value="Undo" @PARENT onclick="vs(event);">'
```

Note too that *onclick”vs(event);”* returns the value of the embedded HTML element in a cell of the on-form report. In the above example the developer can check DBVALUE for either *“Apply”* or *“Undo”* to know which button was clicked.

DesignBais determines for the presence of *@PARENT* by checking that some < (less than) characters precede @PARENT and some > (greater than) characters occur after *@PARENT*. This allows for *@PARENT* to be content in an OFR.

On-form Reports – Allowing more than 10 On-form Reports on a single form.

The variable OUTPUT.REPNUMBER(n) is used to define the report number that each line of OUTPUT.REPORT, OUTPUT.ATTR, OUTPUT.KEYS and DBREPORTROW.ATTR refers to. If the variable is not used, there can only be a maximum of ten on-form-reports used. If this is used, then there is no logical limit.

Note that DBREPORT.UPDATE is implemented for reports using OUTPUT.REPNUMBER.

The report number is defined in the properties of the report control in the forms designer. The main on-form-report arrays are still dimensioned to allow for 10 reports (for backward compatibility). With this modification you can now add a sub-element to the report number. So report number 1 can become 1.1, 1.2, 1.3 etc. This allows for many reports to be defined within the each of the ten subscripts. The OUTPUT.REPNUMBER(n) variable is used to determine which sub-element the line, key or attribute refers to.

Usage:
OUTPUT.REPNUMBER(*Number)<LINE> = Report Number*

Eg.
OUTPUT.REPNUMBER(1)<ROWCOUNT> = “1.4”

If this feature is used each of the OUTPUT.HEADER(n), OUTPUT.WIDTH(n) and OUTPUT.ANCHOR(n) variables are multi-valued for each sub-element. The following example is code that is called by both a standard OFR with an integer process report number, say 3, and a sub element process report number, 9.2 for example.

```
*
INITIALISE.SUBELEMENT.REPORT:
*
      SUBELEMENT.REPORT = 1
      PROCESS.REPORT.NUMBER.ACTUAL = PROCESS.REPORT.NUMBER
      REPORT.SUBELEMENT = FIELD(PROCESS.REPORT.NUMBER,'.',2) + 0
      IF REPORT.SUBELEMENT = 0 THEN
         SUBELEMENT.REPORT = 0
         REPORT.SUBELEMENT = 1
      END
      PROCESS.REPORT.NUMBER = FIELD(PROCESS.REPORT.NUMBER,'.',1)
      IF SUBELEMENT.REPORT THEN
         * Clear only the correct sub element for subelement report 9.2
         JMAX=DCOUNT(OUTPUT.HEADERS(PROCESS.REPORT.NUMBER),AM)
         FOR J=1 TO JMAX
            OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<J,REPORT.SUBELEMENT> = ""
            OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER)<J,REPORT.SUBELEMENT> = ""
            OUTPUT.ANCHOR(PROCESS.REPORT.NUMBER)<J,REPORT.SUBELEMENT> = ""
         NEXT J
         JMAX=DCOUNT(OUTPUT.REPORT(PROCESS.REPORT.NUMBER),AM)
         FOR J=JMAX TO 1 STEP -1
            IF OUTPUT.REPNUMBER(PROCESS.REPORT.NUMBER)<J> = PROCESS.REPORT.NUMBER.ACTUAL THEN
               DEL OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<J>
               DEL OUTPUT.REPNUMBER(PROCESS.REPORT.NUMBER)<J>
               DEL OUTPUT.KEYS(PROCESS.REPORT.NUMBER)<J>
               DEL OUTPUT.ATTR(PROCESS.REPORT.NUMBER)<J>
               DEL OUTPUT.MOUSEOVER(PROCESS.REPORT.NUMBER)<J>
            END
         NEXT J
      END ELSE
          * initialise for report 3
         OUTPUT.REPORT(PROCESS.REPORT.NUMBER) = ""    ; * Variable used to store output in a report format
         OUTPUT.KEYS(PROCESS.REPORT.NUMBER) = ""      ; * Variable used to store key-links to OUTPUT.REPORT
         OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)= ""    ; * Variable used to store report headers
         OUTPUT.ATTR(PROCESS.REPORT.NUMBER) = ""      ; * Variable used to control report attributes
         OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER) = ""    ; * Variable to store widths
         OUTPUT.ANCHOR(PROCESS.REPORT.NUMBER) = ""    ; * Variable to store clickable column numbers
```

```
      OUTPUT.MOUSEOVER(PROCESS.REPORT.NUMBER) = "" ; * Variable to store mouseover style for clickable columns
   END
   RETURN


   BUILD.REPORT:
    *
    * OUTPUT.HEADERS normally 1 attribute per column become multi-valued so for report 9.3 the 3rd value is populated
    *
   OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<1,REPORT.SUBELEMENT> = HEAD1
   OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<2,REPORT.SUBELEMENT> = HEAD2
   OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<3,REPORT.SUBELEMENT> = HEAD3
   OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<4,REPORT.SUBELEMENT> = HEAD4
   OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<5,REPORT.SUBELEMENT> = HEAD5
   OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<6,REPORT.SUBELEMENT> = HEAD6
   OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<7,REPORT.SUBELEMENT> = HEAD7
   *
   DBREPORTHEADER.ATTR<PROCESS.REPORT.NUMBER,1> = "aligncenter"
   DBREPORTHEADER.ATTR<PROCESS.REPORT.NUMBER,2> = "aligncenter"
   DBREPORTHEADER.ATTR<PROCESS.REPORT.NUMBER,3> = "aligncenter"
   DBREPORTHEADER.ATTR<PROCESS.REPORT.NUMBER,4> = "aligncenter"
   DBREPORTHEADER.ATTR<PROCESS.REPORT.NUMBER,5> = "aligncenter"
   DBREPORTHEADER.ATTR<PROCESS.REPORT.NUMBER,6> = "aligncenter"
   DBREPORTHEADER.ATTR<PROCESS.REPORT.NUMBER,7> = "aligncenter"
   *
   * OUTPUT.WIDTHS normally 1 attribute per column become multi-valued so for report 9.3 the 3rd value is populated
   *
   OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER)<1,REPORT.SUBELEMENT> = "2"
   OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER)<2,REPORT.SUBELEMENT> = "8"
   OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER)<3,REPORT.SUBELEMENT> = "8"
   OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER)<4,REPORT.SUBELEMENT> = "8"
   OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER)<5,REPORT.SUBELEMENT> = "4"
   OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER)<6,REPORT.SUBELEMENT> = "4"
   OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER)<7,REPORT.SUBELEMENT> = "8"
   *
   * set RCNT to maximum currently populated attribute position (OUTPUT.REPORT could already contain lines for say
report 9.1)
   *
   RCNT = DCOUNT(OUTPUT.REPORT(PROCESS.REPORT.NUMBER),AM)
   *
   LCNT = 0
   LMAX = DCOUNT(REPORT.ARRAY<1>,VM)
   FOR LL = 1 TO LMAX
      RCNT += 1
      LCNT += 1
      OUTPUT.LINE=LCNT
      OUTPUT.LINE<1,2>=REPORT.ARRAY<1,LL>
      OUTPUT.LINE<1,3>=REPORT.ARRAY<3,LL>
      OUTPUT.LINE<1,4>=REPORT.ARRAY<2,LL>
      OUTPUT.LINE<1,5>=REPORT.ARRAY<4,LL>
      OUTPUT.LINE<1,6>=REPORT.ARRAY<5,LL>
      OUTPUT.LINE<1,7>=REPORT.ARRAY<6,LL>
      *
      OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<RCNT>=OUTPUT.LINE
      *
      * OUTPUT.REPNUMBER is set to 9.3 to flag that this output line belongs to report 9.3 (and not 9.1 say)
      *
      OUTPUT.REPNUMBER(PROCESS.REPORT.NUMBER)<RCNT> = PROCESS.REPORT.NUMBER.ACTUAL
      OUTPUT.ATTR(PROCESS.REPORT.NUMBER)<RCNT>=OUTPUT.AT
      OUTPUT.KEYS(PROCESS.REPORT.NUMBER)<RCNT,2> = ''
      OUTPUT.TITLE(PROCESS.REPORT.NUMBER)<RCNT,2> = REPORT.ARRAY<1,LL>
      OUTPUT.TITLE(PROCESS.REPORT.NUMBER)<RCNT,3> = REPORT.ARRAY<3,LL>
```

```
    OUTPUT.TITLE(PROCESS.REPORT.NUMBER)<RCNT,4> = REPORT.ARRAY<2,LL>
    OUTPUT.MOUSEOVER(PROCESS.REPORT.NUMBER)<RCNT,2> = MO.STYLE
    OUTPUT.MOUSEOVER(PROCESS.REPORT.NUMBER)<RCNT,3> = MO.STYLE
    OUTPUT.MOUSEOVER(PROCESS.REPORT.NUMBER)<RCNT,4> = MO.STYLE
NEXT LL
RETURN
```

OUTPUT.ANCHOR(nn) for a normal on-form report contains a multivalued list of column numbers. For sub-element reports this becomes a subvalued list of column numbers which is multivalued by sub-element. For example:

OUTPUT.ANCHOR(PROCESS.REPORT.NUMBER)<1, REPORT.SUBELEMENT> = 1:SVM:6

On-form Reports – Adding cell titles

See OUTPUT.TITLE(n) above.

On-form Reports – Modifying the container

DBOFRSPEC

The variable DBOFRSPEC is used to specify the properties of the On-form Report container. The default On-form Report container is a grey outline. To change this DBOFRSPEC can be used. The syntax for variable requires three attributes.

Usage:

DBOFRSPEC<1> = "*Report Name*"`: VM : "*Report Name*"
DBOFRSPEC<2> = *"Style Name"* : VM : "*Style Name"*
DBOFRSPEC<3> = "*Style Value"* : VM : "*Style Value*"

Eg.

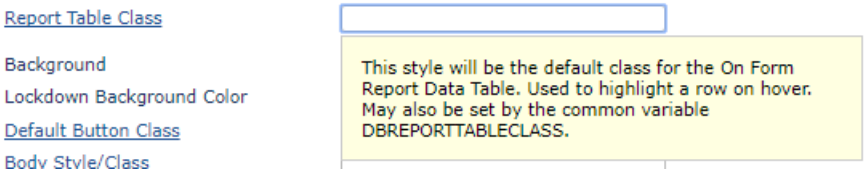
```
DBOFRSPEC<1> = "R.SALES":VM:"R.SALES"
DBOFRSPEC<2> = "borderStyle":VM:"borderColor"
DBOFRSPEC<3> = "solid":VM:"black"
```

On-form Reports – Modifying the Default Row Mouseover appearance

DBREPORTTABLECLASS

The DBREPORTTABLECLASS has 2 attributes and should be used when initially building an On-form Report. It is initialised in DBI.G.INITVARIABLESNET.

The default Report Table Class has been added to the Style Group. This class controls how an On-form Report row appears on mouse hover. The default in the Style Group may be overridden using the DBREPORTTABLECLASS common variable.



DBREPORTTABLECLASS<1> = MV list of report names to which the class in the associated position in attribute 2 applies.
DBREPORTTABLECLASS<2> = MV style/class id from DBISTYLE. If position is blank then no class will apply.

```
*
    PROCESS.REFRESH = "R.REPORT1"
    PROCESS.TYPE = "R"
    DBREPORTTABLECLASS<1> = 'R.DUMMY.REPORT':VM:PROCESS.REFRESH
    DBREPORTTABLECLASS<2> = 'dbaisMouseOverCompare':VM:'dbaisDefaultMouseOver'
    RETURN
```

On-form Reports – Paging Control

Paging control for On-form Reports will prevent the browser from crashing when rendering long On-form Reports. The crash is caused by the buffer size. On Windows the buffer size can be increased as required.

If you need to allow large file uploads/downloads to/from the server, open a CMD shell “as Administrator”, change directory to:

C:\inetpub\adminscripts

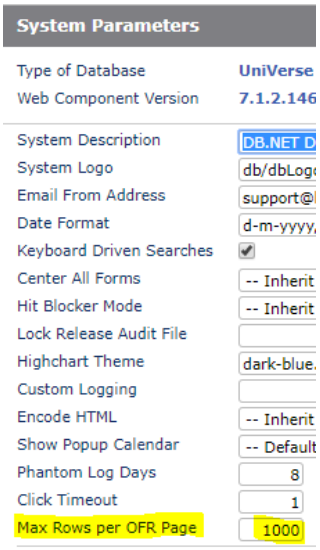
and run the following two commands (in this example the maximum is set to approx. 50 mb):

cscript adsutil.vbs SET w3svc/ASPMaxRequestEntityAllowed 50000000
cscript adsutil.vbs SET w3svc/aspbufferinglimit 50000000

If you do not have the "adminscripts" folder you will need to go to the IIS features and add the management part. The limits are there to guard against cyber attacks and smaller numbers are better. By default, IIS web server allows for limited file size to be uploaded to the web server. For IIS 6 and IIS 7, the default maximum file upload size is 4 MB and 28.6 MB respectively. IIS 7 returns a 404 error (HTTP Error 404.13 - CONTENT_LENGTH_TOO_LARGE) if a user uploads something larger than 30MB. See also IIS Response Buffering Limits in this manual.

There are 2 ways to invoke paging control in On-form Reports.

Set a value in the “Max Rows per OFR Page” parameter in System Parameters.



Setting a large number such as 1000 means that most of the On-form Reports within the DesignBais tools and the application will not page as they will have fewer than 1000 rows. A scrollbar will appear. Large reports, such as the “List all of the field Properties” report for a file with a large number of field property records will page and avoid any potential browser crash.

The Max Rows per OFR Page parameter can be left blank in which case paging will not be invoked. Note that changing the System Parameters value will effect all reports including DesignBais Tools.

You can also set the maximum rows per page in your basic code by setting attribute 2 of the common variable PROCESS.REFRESH as shown in the following example. The maximum number of page rows is set to 40.

```
PROCESS.REFRESH<1,-1> = "R.REVIEW1"
PROCESS.TYPE<1,-1> = "R"
PROCESS.REFRESH<2,-1> = 40
```

The paging control elements are shown in the following figure.

Include Attrib >= 0

Field Name	Screen Label	Attrib	Type	eXpress Reporting	Length	Dec	Just	Conv	Mult	Work	Group
DBIU.PHONE	Contact Phone Number	8	A	No	20		L		N	N	
DBIU.PREFERENCE1	Preference 1	27	A	No	5		L		N	N	
DBIU.PREV.EDIT.FILE	File	49	A	No	30		L		Y	N	PREVEDIT
DBIU.PREV.EDIT.FILE.INPUT.WK	File Name	40	A	N/A	30		L		N	Y	
DBIU.PREV.EDIT.FILE.SEL.WK	File Name	42	A	N/A	30		L		N	Y	
DBIU.PREV.EDIT.FILE.WK	File Name	41	A	N/A	30		L		N	Y	
DBIU.PREV.EDIT.ITEM	Item	50	A	No	30		L		Y	N	PREVEDIT
DBIU.PREV.EDIT.ITEM.INPUT.WK	Item	37	A	N/A	30		L		N	Y	
DBIU.PREV.EDIT.ITEM.SEL.WK	Item	39	A	N/A	30		L		N	Y	
DBIU.PREV.EDIT.ITEM.WK	Item	38	A	N/A	30		L		N	Y	
DBIU.PREV.EDIT.PAGEROWS.WK	Rows per Page	43	N	N/A	10		R		N	Y	
DBIU.PRINTERS	Printer Names	12	A	No	30		L		Y	N	
DBIU.REP.DISPLAV1.WK	Misc Display Field 1	30	A	N/A	30		L		N	Y	
DBIU.REP.DISPLAV2.WK	Misc Display Field 2	31	A	N/A	30		L		N	Y	
DBIU.REP.DISPLAV3.WK	Misc Display Field 3	32	A	N/A	30		L		N	Y	
DBIU.SET.ACCOUNT.WK	Account or Account Path	15	A	N/A	30		L		Y	Y	SETACCOUNT
DBIU.SET.START.FORM.WK	Start Form	16	A	N/A	30		L		Y	Y	SETACCOUNT
DBIU.SET.START.PROCESS.WK	Start Process (Form Load)	17	A	N/A	20		L		Y	Y	SETACCOUNT
DBIU.SET.USER.WK	User Name	2	A	N/A	12		L		Y	Y	SETUSER
DBIU.SHOW.STATUS	Show Status Bar	47	A	No	3		L		N	N	
DBIU.START.FORM	Start Form	16	A	No	20		L		Y	N	ACCOUNTS
DBIU.START.PROCESS	Start Process (Form Load)	17	A	No	20		L		Y	N	ACCOUNTS

Total Rows 74 Page 3 / 4

If paging is active then:

Report width >= 64 pixels then paging control included
Report width >= 96 pixels then paging control + step control included
Report width >= 221 pixels then paging control + step control + Go To Page included
Report width >= 321 pixels then paging control + step control + Go To Page + Total Rows included

Paging Control parameters are available on the form style group. Refer to Style Group

File Variables in DesignBais Common

The following DesignBais file variables are defined in DBI.COMMON

1. F.DBIFILES
2. F.DBIFORMHTML
3. F.DBIFORMS
4. F.DBIGLOBAL
5. F.DBIGROUPS
6. F.DBILICENCE
7. F.DBIMENUS
8. F.DBIPARMS
9. F.DBIPROP
10. F.DBIREPORTS
11. F.DBIRULE
12. F.DBISELECT
13. F.DBISESSIONS
14. F.DBISESSIONS.MAIN
15. F.DBISTATS
16. F.DBISTYLE
17. F.DBISTYLEGROUP
18. F.DBISYSFILES
19. F.DBISYSFORMS
20. F.DBISYSPROP
21. F.DBISYSSELECT
22. F.DBIUSERLOG
23. F.DBIUSERS

The following DesignBais files are not defined in DBI.COMMON. These files must be opened in your basic code before you reference them:

1. DBIAUDIT
2. DBIAUDIT.EXT
3. DBIBACKUP
4. DBICHK
5. DBICHKEXTRACT
6. DBICON
7. DBIEOP
8. DBIGLOSSARY
9. DBIHELP
10. DBIPMCODE
11. DBIPMLINK
12. DBIPMPROC
13. DBIPMRULE
14. DBIPMSTATUS
15. DBIPMSTEP
16. DBIXMLLOG

Displaying Graphs on DesignBais forms

DesignBais allows the developer to produce a number of high quality graphs on a form. Graphs can appear on base and modal forms.

From Version 7 onwards graph elements on a form, both existing (created pre-version 7/8) and new (created with version 7/8), will be rendered using the Highchart charting component. Note that your basic code must follow the pattern defined for graphs, not highcharts, if you use the graph form element in version 7 or later. We recommend that highchart form elements and matching basic code be used in version 7 and later versions.

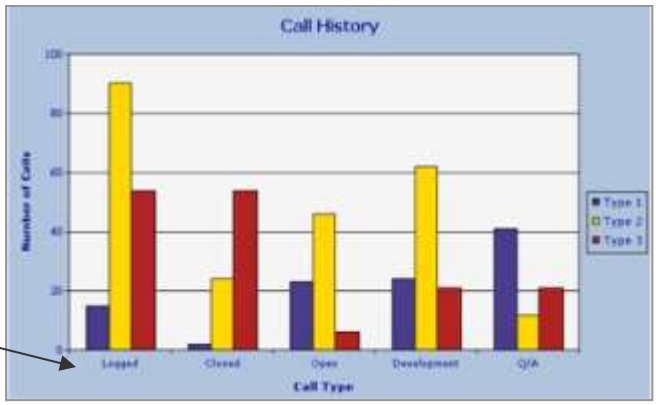
Graphs require the Office Web Component (OWC10) to be installed on the web server. See web component installation instructions for details.

Up to 10 graphs, and/or Highcharts, can be placed on an individual form. If On-form Reports exist on the form then this will reduce the number of graphs that can be used. You can use OUTPUT.REPNUMBER(n) to extend the number of reports on a form but this feature is not available for graphs and highcharts. If report numbers 1 to 4 are used for reports, then only 6 report number slots remain for graphs and highcharts. The 4 slots for reports can of course still use OUTPUT.REPNUMBER(n) to extend the number of reports.

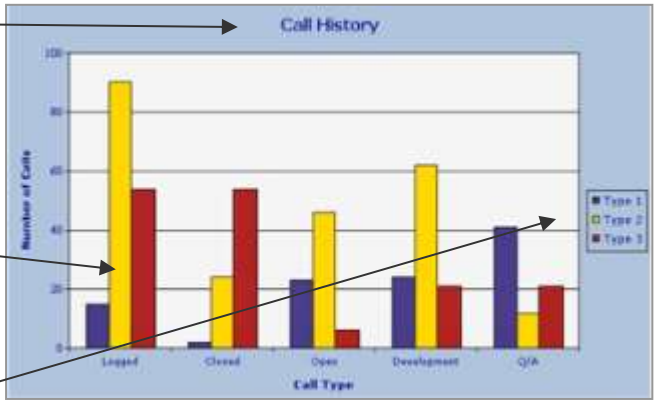
Graphs are created in a similar fashion to On-form Reports. Below is an example of a program to create a graph on a form and an example of the graph produced.

Note: Where VM is used is represents a Value Mark, @VM char(254)

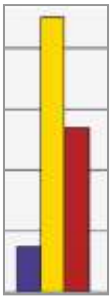
```
*
REPORT.NUMBER = 2

OUTPUT.REPORT(REPORT.NUMBER) = "" ; * Variable used to store output in a graph format
OUTPUT.KEYS(REPORT.NUMBER)   = "" ; * Not used by Graphs
OUTPUT.HEADERS(REPORT.NUMBER)= "" ; * Variable used to store Graph headers
OUTPUT.ATTR(REPORT.NUMBER)   = "" ; * Variable used to control Graph Attributes

OUTPUT.AT    = ''     ; Attributes for the graph

*
* Build the Headers, These will be the data headings
*
OUTPUT.HEADERS(REPORT.NUMBER)<1> = "Logged"
OUTPUT.HEADERS(REPORT.NUMBER)<2> = "Closed"
OUTPUT.HEADERS(REPORT.NUMBER)<3> = "Open"
OUTPUT.HEADERS(REPORT.NUMBER)<4> = "Development"
OUTPUT.HEADERS(REPORT.NUMBER)<5> = "Q/A"
```




```
*
* Graph Title
*
OUTPUT.AT<1> = "Call History" ;* Graph Title
```




```
*
* Define the Type of Chart for Each Series
*
* Type of Graph (All 3 Series are bar chart type 0)
OUTPUT.AT<2> = 0:VM:0:VM:0  ;

*
* Define the Legend Descriptions
*

 OUTPUT.AT<3> = "Type 1":VM:"Type 2":VM:"Type 3" ;
* Legend for the 3 Graph Series


*
* Define the Series Colors
*
* Colors for the 3 Graph Series
OUTPUT.AT<4> = "darkslateblue":VM:"gold":VM:"firebrick"
```



*** Colors can be any of the 140 browser compliant colors or RGB colors in the form #RRGGBB (Eg. #E4A4E4)**

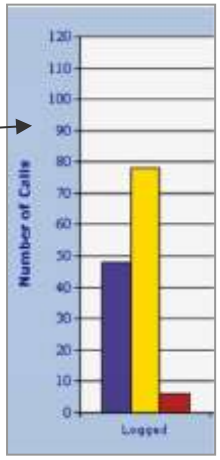
*

*** Scale - Change the scale being used.** The above examples use the default scale.

*

* Changing the scale to 10 provides the following result.

```
OUTPUT.AT<5> = 10 ;* Scale
```

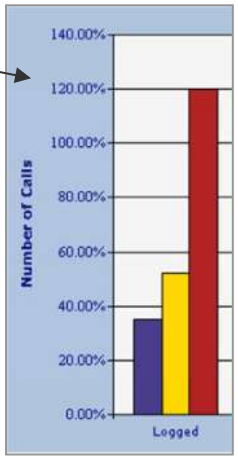


*

* **Number Format - Change the number format on the 'Y' axis.**

*

```
OUTPUT.AT<5> = '0.2'    ;* Scale of 0.2
OUTPUT.AT<6> = "0.00%"  ;* Number format as a percentage and 2 decimal places
```



*

*** Titles – This Section controls the Title style for the chart**.

*

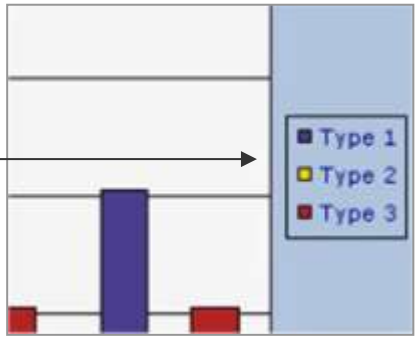
```
OUTPUT.AT<7> = 1 ; * Has a Title
* [Blank True, 0 No Title]
OUTPUT.AT<8> = "12" ; * Title Font Size [Blank Default]
OUTPUT.AT<9> = "Navy" ; * Title Color [Default Black]
OUTPUT.AT<10> = "Verdana" ; * Title Font
OUTPUT.AT<11> = 1 ; * Font Bold-1 = Bold, 0-Normal [Blank Normal]
```

*

* **Legend – This Section control the Legend Style for the chart**

*

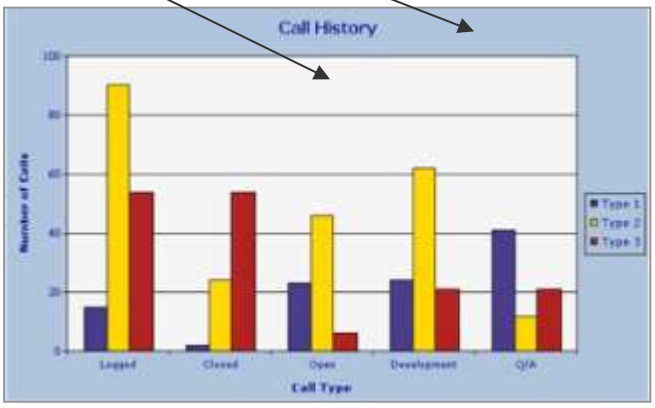
```
OUTPUT.AT<12> = 1 ;* Has Legend [Blank True, 0 No Legend]
OUTPUT.AT<13> = '8' ; * Legend Font Size [Blank Default font size]
OUTPUT.AT<14> = 'navy' ; * Legend Font Color
*                          [Blank Default font color]
OUTPUT.AT<15> = 'Lightsteelblue' ; * Legend Interior Color
*                                  [Blank White]
OUTPUT.AT<16> = '4' ; * Legend Position,
*     1-Top, 2-Bottom, 3-Left, 4-Right [Blank Right]
OUTPUT.AT<17> = "Verdana" ;* Legend Font [Blank Default font]
OUTPUT.AT<18> = 0 ; * Font, 1-Bold, 0-Normal [Blank Normal]
```



Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

*
*** BackGround Color – This Section controls the main chart background color and the main interior color for the graph.**
*

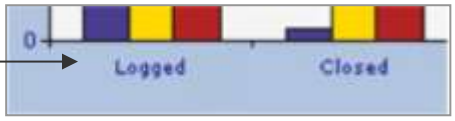
```
OUTPUT.AT<19> = "Lightsteelblue" ; * Main BackGround [Blank Default Background Color]
OUTPUT.AT<20> = "whitesmoke"   ; * Main Interior Color [Blank Default Interior Color]
```



*
*** X Axis Title – This Section controls the style and content of the X Axis**
*


```
OUTPUT.AT<21> = "Call Type" ; * X Axis Title
*                             [Blank No X Axis Title]
OUTPUT.AT<22> = "8" ;* X Axis Title Font Size
OUTPUT.AT<23> = 'navy' ; * X Axis Title Color
OUTPUT.AT<24> = 'Verdana' ; * X Axis Title Font
OUTPUT.AT<25> = 1 ; * X Axis Font Title Weight
```

*
*** X Axis Labels – This Sections controls the display style of the X Axis Labels**
*

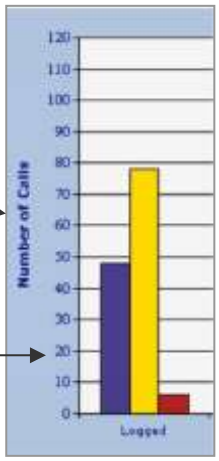



```
OUTPUT.AT<26> = "7" ;* X Axis Label Font Size
OUTPUT.AT<27> = 'navy' ;* X Axis Label Font Color
OUTPUT.AT<28> = "Verdana" ; * X Axis Label Font
OUTPUT.AT<29> = 0 ; * X Axis Label Font Bold
```

*
*** Y Axis Title – This Section controls the style and content of the Y Axis**

*

```
OUTPUT.AT<30> = "Number of Calls" ; * Y Axis Title [Blank No Y Axis Title]
OUTPUT.AT<31> = "8" ;* Y Axis Font Size
OUTPUT.AT<32> = 'navy' ; * Y Axis Color
OUTPUT.AT<33> = 'Verdana' ; * Y Axis Font
OUTPUT.AT<34> = 1 ; * Y Axis Font Weight
```



*
* **Y Axis Labels – This Sections controls the display style of the Y Axis Labels**
*

```
OUTPUT.AT<35> = "8" ;* Y Axis Label Font Size
OUTPUT.AT<36> = 'navy' ;* Y Axis Label Font Color
OUTPUT.AT<37> = "Times New Roman" ; * Y Axis Label Font
OUTPUT.AT<38> = 0 ; * X Axis Label Font Bold
```

*
* **Assign the OUTPUT.AT to the Graph attribute variable**
*

```
OUTPUT.ATTR(REPORT.NUMBER) = OUTPUT.AT ; * Assign Attributes to the Graph variable
```

*
* **Build the Data for the Graph**
*
* If the following Example the data is populated in the 3 series for each of the 5 headings

```
   FOR LL = 1 TO 3
      FOR DL = 1 TO 5
         OUTPUT.REPORT(REPORT.NUMBER)<LL,DL> = (LL*RND(50)) ; * Provide Data
      NEXT DL
   NEXT LL
```

*
*
* **Now set the refresh parameters to inform the DesignBais Hub to redraw the graph**
*

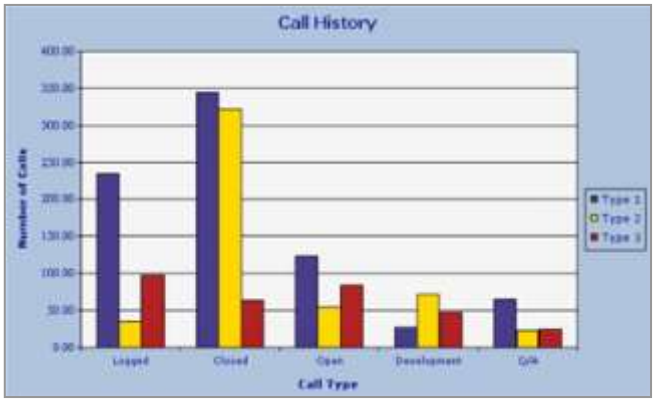
```
PROCESS.TYPE = "G"
PROCESS.REFRESH = "G.GRAPH2"
```

Example of graph populated with data

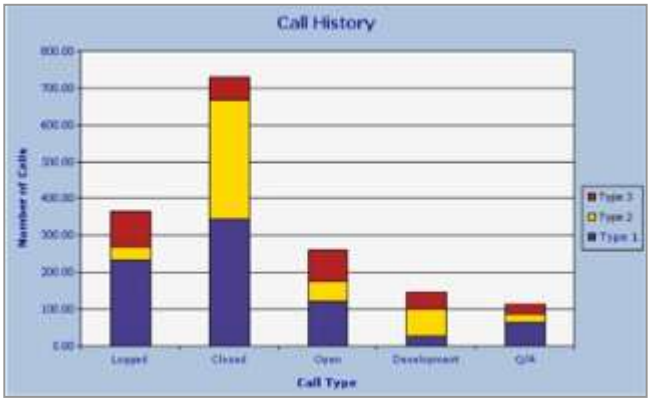
	Type 1	Type 2	Type 3
Logged	234	35	98
Closed	345	321	64
Open	123	54	84
Development	27	72	48
Q/A	65	23	25

In the above example there are three Series, Data Type 1, Data Type 2 and Data Type 3.
There are Five data elements for each series, Logged, Closed, Open, Development and Q/A

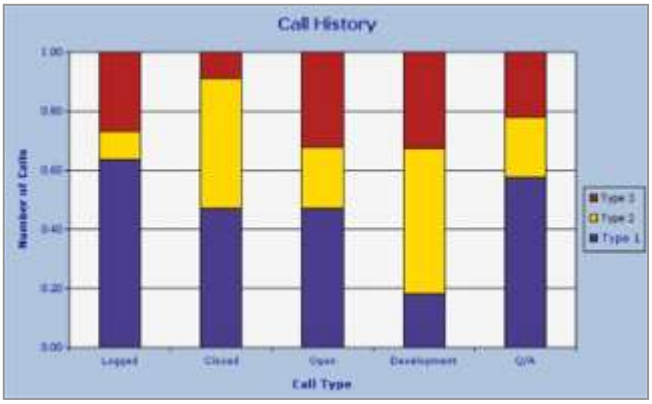
The resultant chart as a Series Type 0.



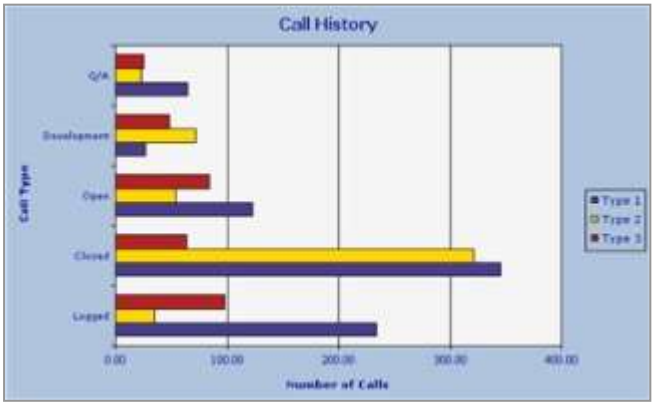
Series Type 1.



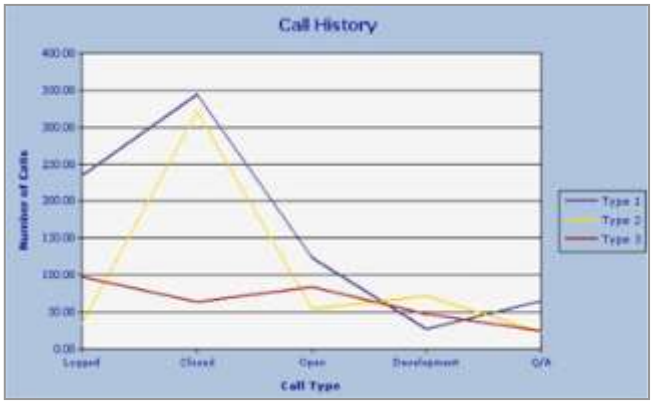
Series Type 2.



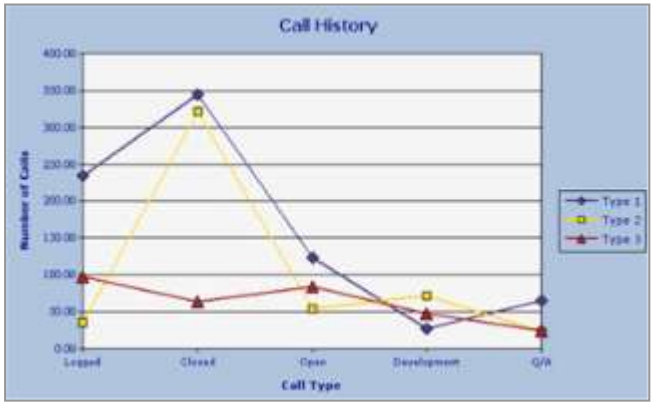
Series Type 3.



Series Type 7

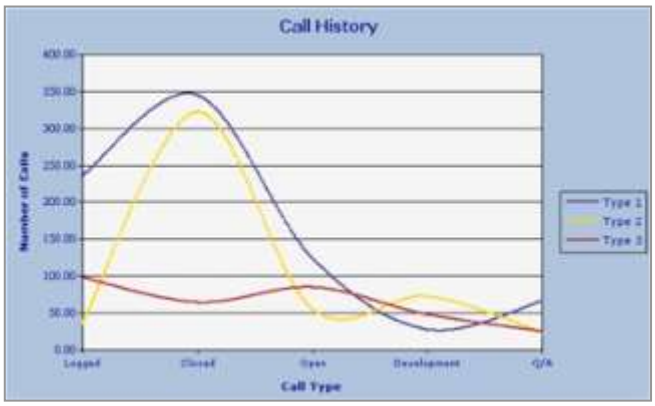


Series Type 9

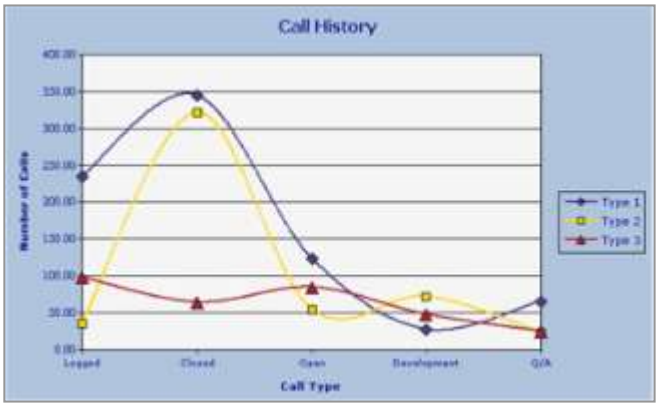


Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

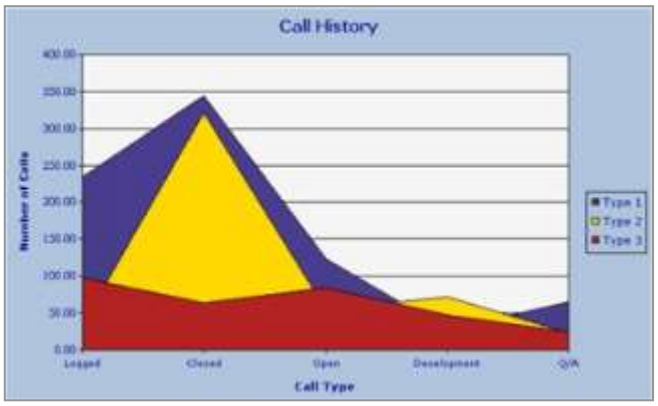
Series Type 13



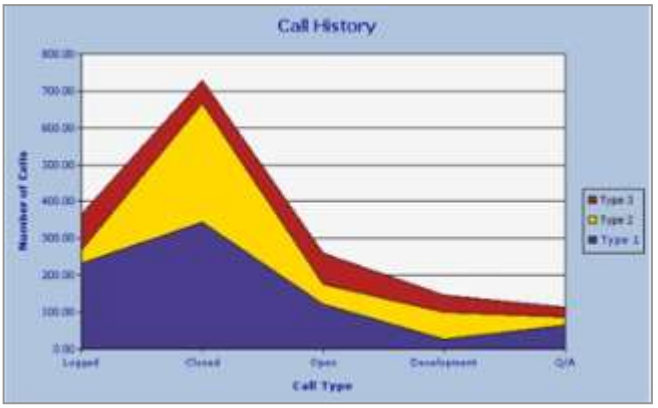
Series Type 15



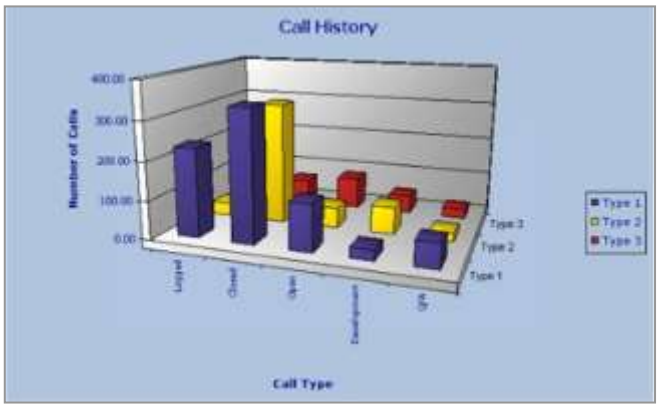
Series Type 29



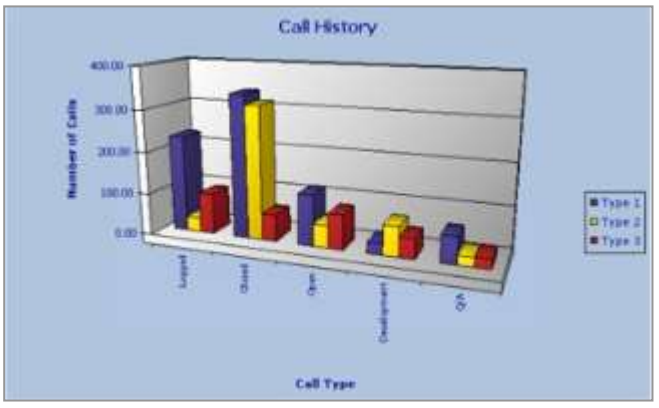
Series Type 30



Series Type 46

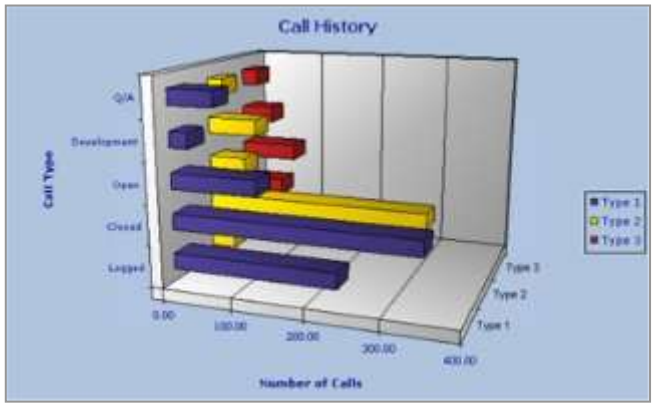


Series Type 47

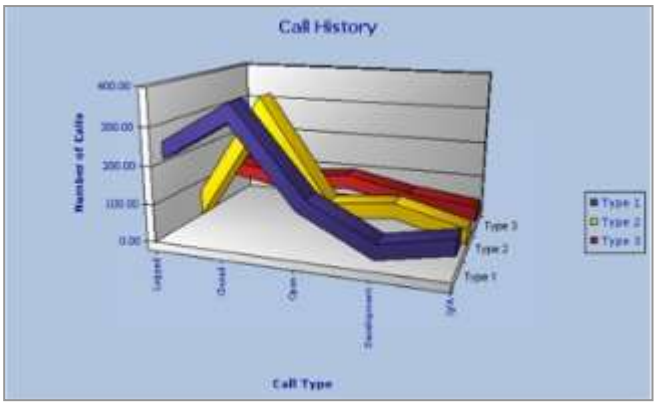


Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

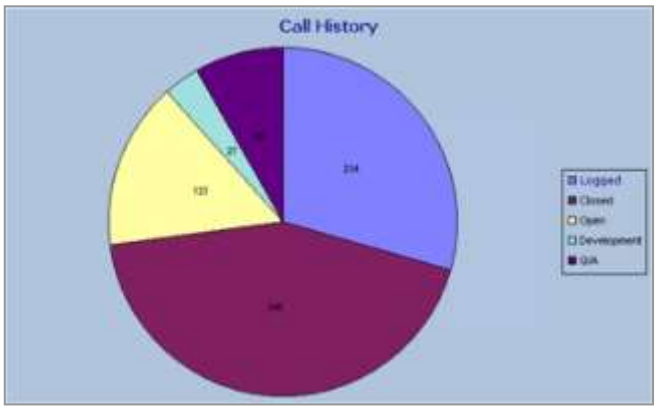
Series Type 50



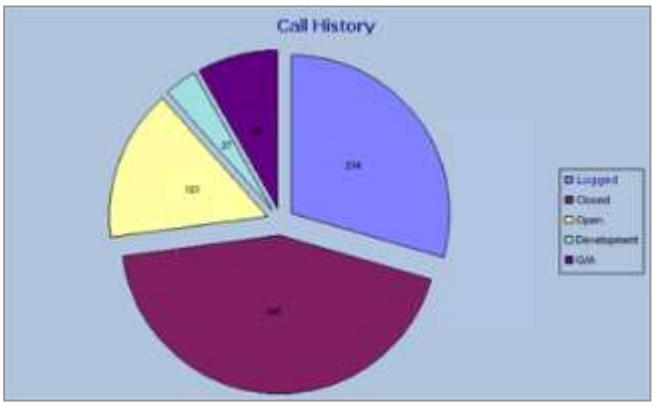
Series Type 54



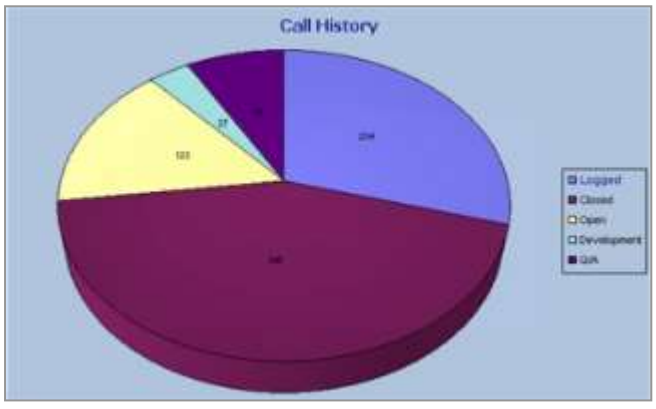
Series Type 18



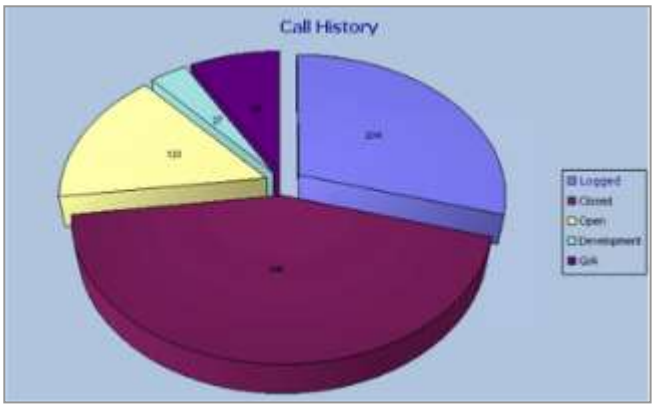
Series Type 20



Series Type 58



Series Type 59



Creating Images from a graph

It is possible to add an image name to the attributes of a graph definition in DesignBais releases prior to Release 7.

Eg.

```
OUTPUT.AT<40> = "myImageID.gif" ;* If left blank, no image is saved. Example: OUTPUT.AT<40> = 'images/db/dbSplash.jpg'
```

This will save the image of the graph to images\dbgraphs directory.

*** If you intend to use this method, you must ensure that the images\dbgraphs directory is present ***

If you want to ensure that the graph is always updated you should use a unique image name. Otherwise, the browser will cache the image and the Graph image will not change.

You can then add the image name to a report. To ensure that the image name is dynamic, use DBIMAGESPEC common variable.

Eg.

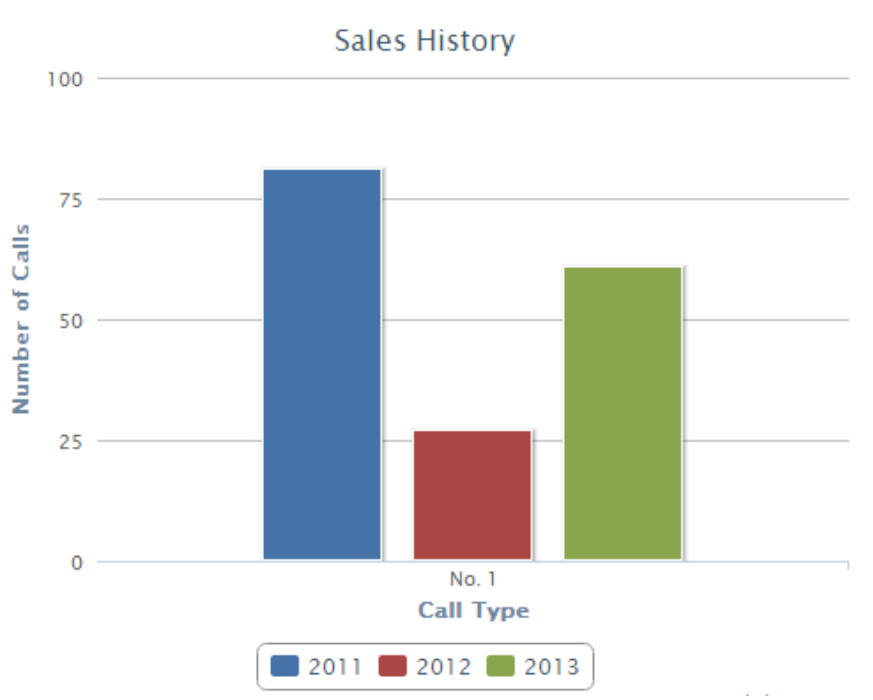
```
DBIMAGESPEC<1> = "myblankgraph.gif"
DBIMAGESPEC<2> = DBKEY:".gif"
```

Displaying Highcharts on DesignBais forms

DesignBais supports the Highchart charting component. There is a default Highchart theme which will be used by DesignBais unless a theme is defined in System Parameters or in Global Parameters. The System Parameters entry will override the Global Parameters entry. When changing of adding a theme it is necessary to refresh the browser for the change to take effect.

We have introduced support for the following chart types:

Column



Code to create the Highchart:

For compatibility with the OWC charts, we have kept the functions of a number of attributes in OUTPUT.ATTR

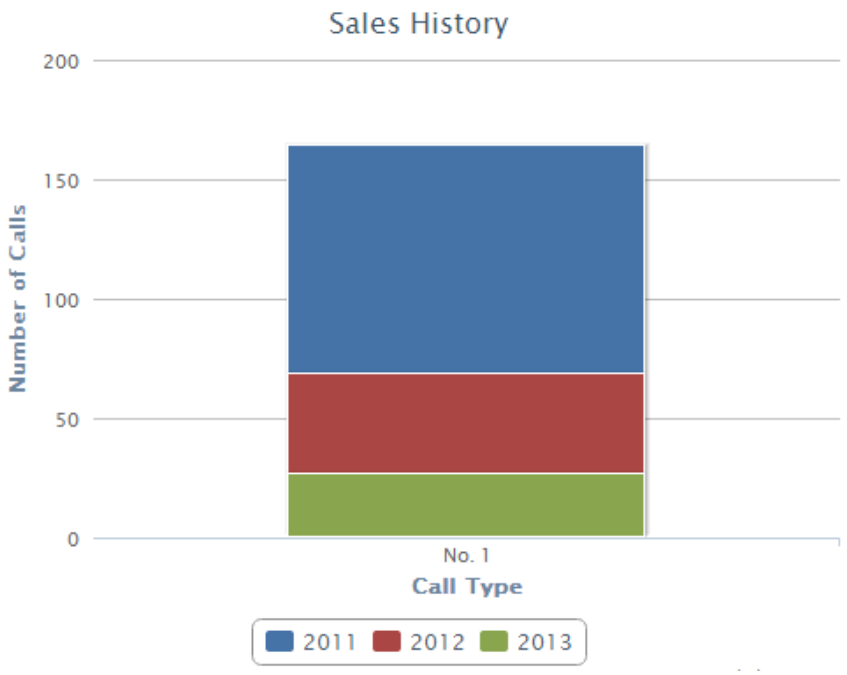
A number of attributes are not required. Below is an example of the code to build the chart.

```
BUILD.HICHART:
*
      PROCESS.REPORT.NUMBER = 2

      OUTPUT.REPORT(PROCESS.REPORT.NUMBER) = ""    ; * Variable used to store output in a report format
      OUTPUT.KEYS(PROCESS.REPORT.NUMBER) = ""      ; * Variable used to store key-links to OUTPUT.REPORT
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)= ""    ; * Variable used to store report headers
      OUTPUT.ATTR(PROCESS.REPORT.NUMBER) = ""      ; * Variable used to control report attributes
      OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER) = ""    ; * Variable to store widths
      OUTPUT.ANCHOR(PROCESS.REPORT.NUMBER) = 0
      OUTPUT.LINE = ''
      OUTPUT.AT = ''

      FOR DL = 1 TO 3
         OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<DL> = 'No. ':DL
      NEXT DL
*

      OUTPUT.AT = "Sales History"
*
      OUTPUT.AT<2> = 'column':VM:'column':VM:'column'
      OUTPUT.AT<3> = "2011":VM:"2012":VM:"2013"
*
* Legend
*
      OUTPUT.AT<12> = 1                  ; * Has Legend [Blank True, 0 No Legend]
      OUTPUT.AT<16,1> = 'bottom'         ; * [bottom] / top / left / right
      OUTPUT.AT<16,2> = 'horizontal'     ; * [horizontal] / vertical
      OUTPUT.AT<16,3> = 'center'         ; * [center] / right / left
      OUTPUT.AT<16,4> = 1                ; * borderwidth in pixels
*
* BackGround COlor
*
      OUTPUT.AT<21> = "Call Type"        ; * X Axis Title [Blank No X Asis Title]
*
      OUTPUT.AT<30> = "Number of Calls"  ; * Y Axis Title [Blank No Y Asis Title]
      OUTPUT.AT<39> = 1           ; * Add colorByPoint to one graph/series
*
      OUTPUT.ATTR(PROCESS.REPORT.NUMBER) = OUTPUT.AT
*
      DD = ""

      FOR LL = 1 TO 1
         FOR DL = 1 TO 4
            OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<LL,DL> = RND(100)        ; * Provide Data
         NEXT DL
      NEXT LL
*
      PROCESS.TYPE<1,-1> = "H"
      PROCESS.REFRESH<1,-1> = "H.HICHART2"
*
      RETURN
```

Column, stacked

Changing the first series type attribute to stacked will create a stacked column.

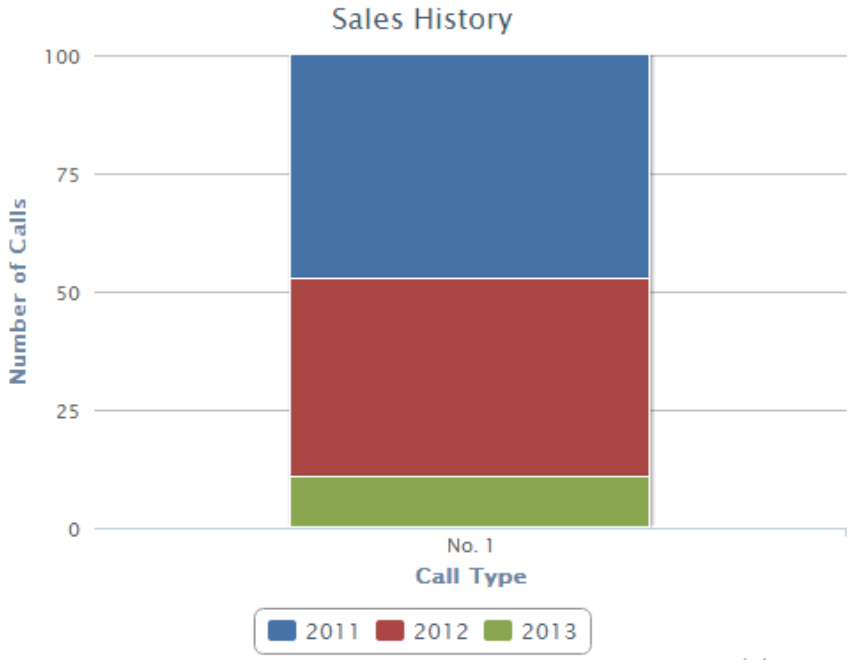
```
OUTPUT.AT<2> = 'column,stacked':VM:'column':VM:'column'
```



Column, stacked, percent

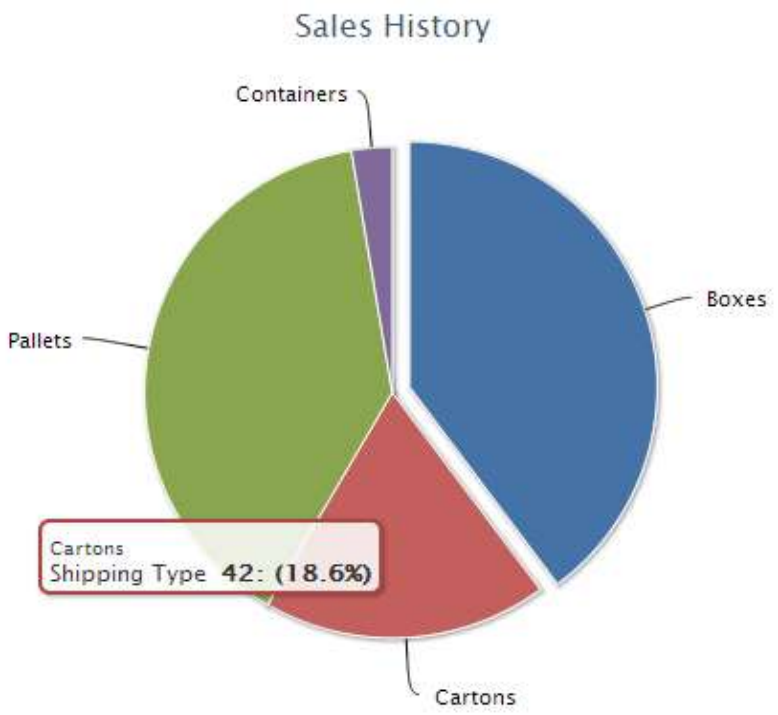
Changing the first series type attribute to stacked,percent will create a stacked percentage column.

```
OUTPUT.AT<2> = 'column,stacked,percent':VM:'column':VM:'column'
```



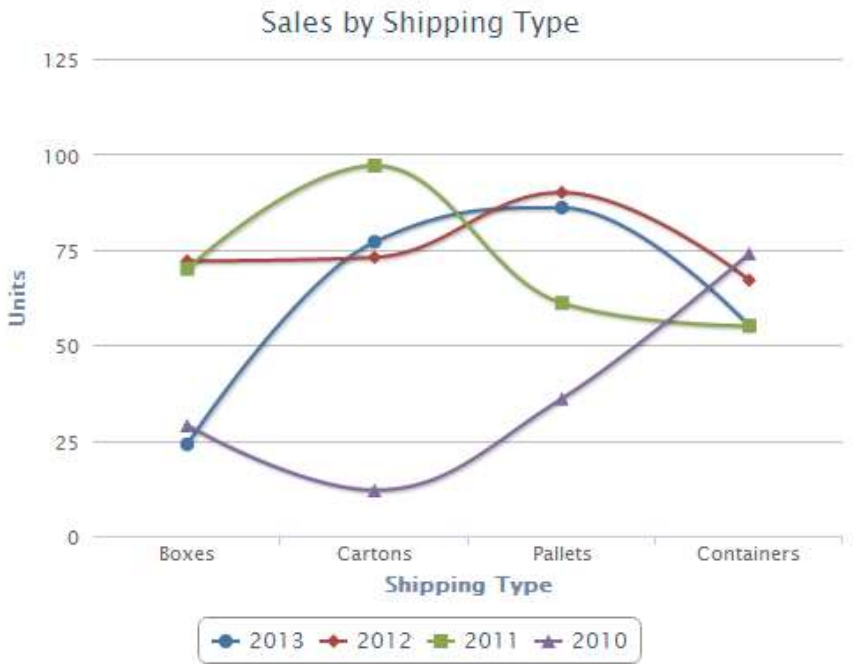
Pie Chart

Highchart pie charts can be colored. Colors are multivalued in OUTPUT.ATTR(REPORT.SUBSCRIPT)<4>.




```
BUILD.HICHART:
*
      PROCESS.REPORT.NUMBER = 2

      OUTPUT.REPORT(PROCESS.REPORT.NUMBER) = ""    ; * Variable used to store output in a report format
      OUTPUT.KEYS(PROCESS.REPORT.NUMBER) = ""      ; * Variable used to store key-links to OUTPUT.REPORT
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)= ""    ; * Variable used to store report headers
      OUTPUT.ATTR(PROCESS.REPORT.NUMBER) = ""      ; * Variable used to control report attributes
      OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER) = ""    ; * Variable to store widths
      OUTPUT.ANCHOR(PROCESS.REPORT.NUMBER) = 0
      OUTPUT.LINE = ''
      OUTPUT.AT = ''

      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<1> = "Boxes"
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<2> = "Cartons"
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<3> = "Pallets"
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<4> = "Containers"
*

      OUTPUT.AT = "Sales by Shipping Type"
*
      OUTPUT.AT<2> = 'pie'
      OUTPUT.AT<3> = "2012"
*
* Legend
*
      OUTPUT.AT<12> = 1                  ; * Has Legend [Blank True, 0 No Legend]
      OUTPUT.AT<16,1> = 'bottom'         ; * [bottom] / top / left / right
      OUTPUT.AT<16,2> = 'horizontal'     ; * [horizontal] / vertical
      OUTPUT.AT<16,3> = 'center'         ; * [center] / right / left
      OUTPUT.AT<16,4> = 1                ; * borderwidth in pixels
*
* BackGround COlor
*
      OUTPUT.AT<21> = "Shipping Type"        ; * X Axis Title [Blank No X Asis Title]
*
      OUTPUT.AT<30> = "Number of Calls"  ; * Y Axis Title [Blank No Y Asis Title]
*
      OUTPUT.ATTR(PROCESS.REPORT.NUMBER) = OUTPUT.AT
*
      DD = ""

      FOR LL = 1 TO 4
         OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<LL> = RND(100)        ; * Provide Data
      NEXT LL
*
      PROCESS.TYPE<1,-1> = "H"
      PROCESS.REFRESH<1,-1> = "H.HICHART2"
*
      RETURN
```

SPLine Chart

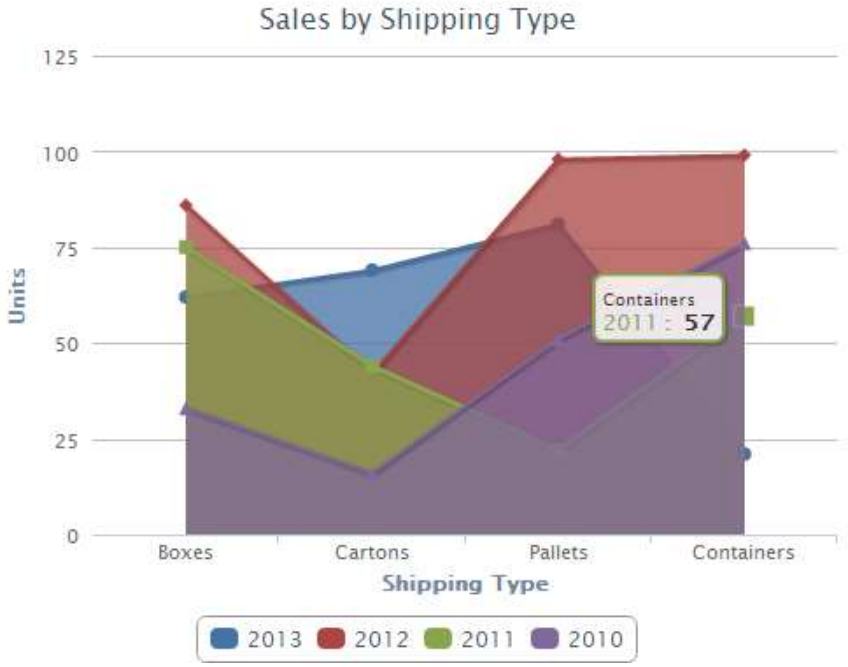



```
BUILD.HICHART:
*
      PROCESS.REPORT.NUMBER = 2

      OUTPUT.REPORT(PROCESS.REPORT.NUMBER) = ""    ; * Variable used to store output in a report format
      OUTPUT.KEYS(PROCESS.REPORT.NUMBER) = ""      ; * Variable used to store key-links to OUTPUT.REPORT
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)= ""    ; * Variable used to store report headers
      OUTPUT.ATTR(PROCESS.REPORT.NUMBER) = ""      ; * Variable used to control report attributes
      OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER) = ""    ; * Variable to store widths
      OUTPUT.ANCHOR(PROCESS.REPORT.NUMBER) = 0
      OUTPUT.LINE = ''
      OUTPUT.AT = ''

      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<1> = "Boxes"
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<2> = "Cartons"
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<3> = "Pallets"
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<4> = "Containers"
*

      OUTPUT.AT = "Sales by Shipping Type"
*
      OUTPUT.AT<2> = 'spline' : VM : 'spline' : VM : 'spline' : VM : 'spline'
      OUTPUT.AT<3> = "2013" : VM : "2012" : VM : "2011" : VM : "2010"
*
* Legend
*
      OUTPUT.AT<12> = 1                  ; * Has Legend [Blank True, 0 No Legend]
      OUTPUT.AT<16,1> = 'bottom'         ; * [bottom] / top / left / right
      OUTPUT.AT<16,2> = 'horizontal'     ; * [horizontal] / vertical
      OUTPUT.AT<16,3> = 'center'         ; * [center] / right / left
      OUTPUT.AT<16,4> = 1                ; * borderwidth in pixels
*
* BackGround COlor
*
      OUTPUT.AT<21> = "Shipping Type"        ; * X Axis Title [Blank No X Asis Title]
*
      OUTPUT.AT<30> = "Units"  ; * Y Axis Title [Blank No Y Asis Title]
*
      OUTPUT.ATTR(PROCESS.REPORT.NUMBER) = OUTPUT.AT
*
      DD = ""

      FOR LL = 1 TO 4
         FOR RR = 1 TO 4
            OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<LL,RR> = RND(100)        ; * Provide Data
         NEXT RR
      NEXT LL
*
      PROCESS.TYPE<1,-1> = "H"
      PROCESS.REFRESH<1,-1> = "H.HICHART2"
*
      RETURN
```

Line Chart

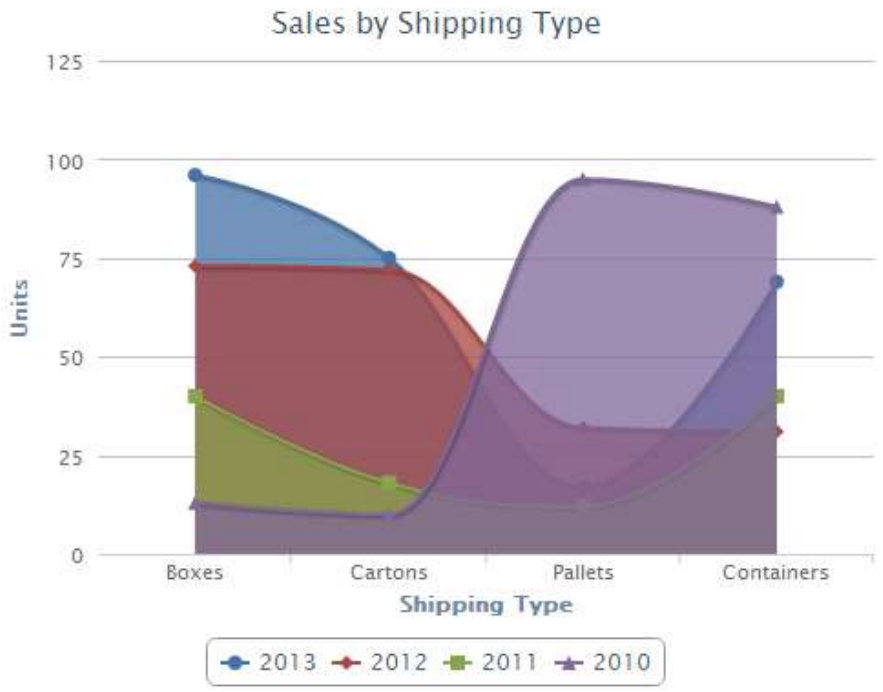

```
BUILD.HICHART:
*
      PROCESS.REPORT.NUMBER = 2

      OUTPUT.REPORT(PROCESS.REPORT.NUMBER) = ""    ; * Variable used to store output in a report format
      OUTPUT.KEYS(PROCESS.REPORT.NUMBER) = ""      ; * Variable used to store key-links to OUTPUT.REPORT
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)= ""    ; * Variable used to store report headers
      OUTPUT.ATTR(PROCESS.REPORT.NUMBER) = ""      ; * Variable used to control report attributes
      OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER) = ""    ; * Variable to store widths
      OUTPUT.ANCHOR(PROCESS.REPORT.NUMBER) = 0
      OUTPUT.LINE = ''
      OUTPUT.AT = ''

      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<1> = "Boxes"
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<2> = "Cartons"
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<3> = "Pallets"
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<4> = "Containers"
*

      OUTPUT.AT = "Sales by Shipping Type"
*
      OUTPUT.AT<2> = 'line' : VM : 'line' : VM : 'line' : VM : 'line'
      OUTPUT.AT<3> = "2013" : VM : "2012" : VM : "2011" : VM : "2010"
*
* Legend
*
      OUTPUT.AT<12> = 1                  ; * Has Legend [Blank True, 0 No Legend]
      OUTPUT.AT<16,1> = 'bottom'         ; * [bottom] / top / left / right
      OUTPUT.AT<16,2> = 'horizontal'     ; * [horizontal] / vertical
      OUTPUT.AT<16,3> = 'center'         ; * [center] / right / left
      OUTPUT.AT<16,4> = 1                ; * borderwidth in pixels
*
* BackGround COlor
*
      OUTPUT.AT<21> = "Shipping Type"        ; * X Axis Title [Blank No X Asis Title]
*
      OUTPUT.AT<30> = "Units"  ; * Y Axis Title [Blank No Y Asis Title]
*
      OUTPUT.ATTR(PROCESS.REPORT.NUMBER) = OUTPUT.AT
*
      DD = ""

      FOR LL = 1 TO 4
         FOR RR = 1 TO 4
            OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<LL,RR> = RND(100)        ; * Provide Data
         NEXT RR
      NEXT LL
*
      PROCESS.TYPE<1,-1> = "H"
      PROCESS.REFRESH<1,-1> = "H.HICHART2"
*
      RETURN
```

Area Chart



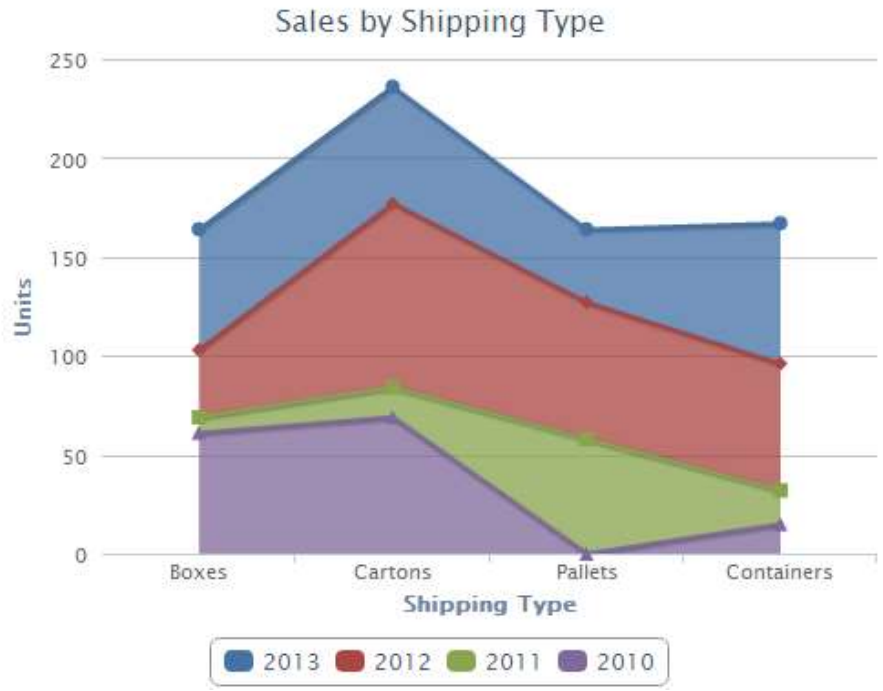

```
BUILD.HICHART:
*
      PROCESS.REPORT.NUMBER = 2

      OUTPUT.REPORT(PROCESS.REPORT.NUMBER) = ""    ; * Variable used to store output in a report format
      OUTPUT.KEYS(PROCESS.REPORT.NUMBER) = ""      ; * Variable used to store key-links to OUTPUT.REPORT
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)= ""    ; * Variable used to store report headers
      OUTPUT.ATTR(PROCESS.REPORT.NUMBER) = ""      ; * Variable used to control report attributes
      OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER) = ""    ; * Variable to store widths
      OUTPUT.ANCHOR(PROCESS.REPORT.NUMBER) = 0
      OUTPUT.LINE = ''
      OUTPUT.AT = ''

      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<1> = "Boxes"
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<2> = "Cartons"
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<3> = "Pallets"
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<4> = "Containers"
*

      OUTPUT.AT = "Sales by Shipping Type"
*
      OUTPUT.AT<2> = 'area' : VM : 'area' : VM : 'area' : VM : 'area'
      OUTPUT.AT<3> = "2013" : VM : "2012" : VM : "2011" : VM : "2010"
*
* Legend
*
      OUTPUT.AT<12> = 1                  ; * Has Legend [Blank True, 0 No Legend]
      OUTPUT.AT<16,1> = 'bottom'         ; * [bottom] / top / left / right
      OUTPUT.AT<16,2> = 'horizontal'     ; * [horizontal] / vertical
      OUTPUT.AT<16,3> = 'center'         ; * [center] / right / left
      OUTPUT.AT<16,4> = 1                ; * borderwidth in pixels
*
* BackGround COlor
*
      OUTPUT.AT<21> = "Shipping Type"        ; * X Axis Title [Blank No X Asis Title]
*
      OUTPUT.AT<30> = "Units"  ; * Y Axis Title [Blank No Y Asis Title]
*
      OUTPUT.ATTR(PROCESS.REPORT.NUMBER) = OUTPUT.AT
*
      DD = ""

      FOR LL = 1 TO 4
         FOR RR = 1 TO 4
            OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<LL,RR> = RND(100)        ; * Provide Data
         NEXT RR
      NEXT LL
*
      PROCESS.TYPE<1,-1> = "H"
      PROCESS.REFRESH<1,-1> = "H.HICHART2"
*
      RETURN
```

AreaSPLine Chart



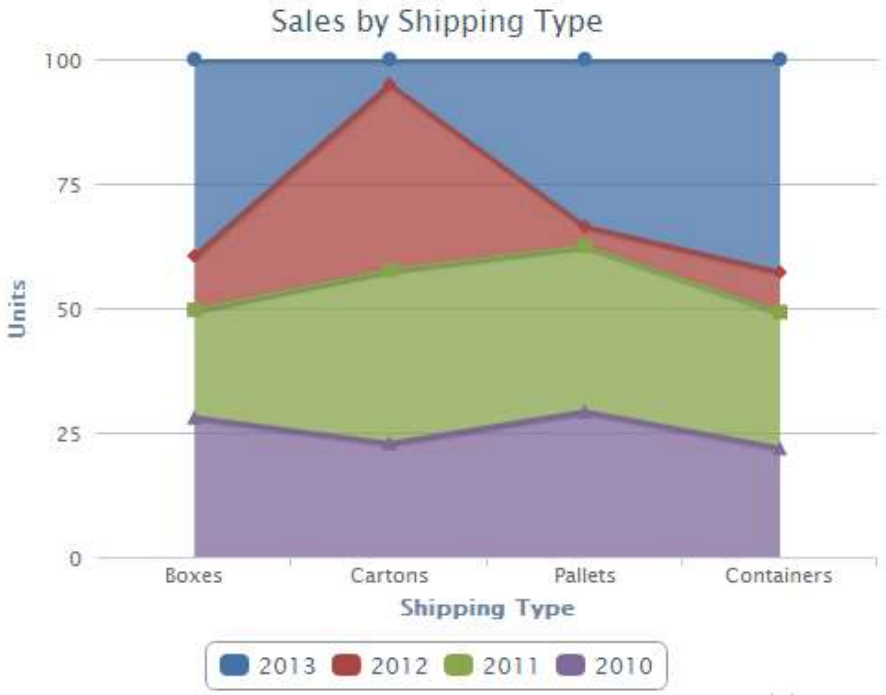

```
OUTPUT.AT<2> = 'areaspline' : VM : 'areaspline' : VM : 'areaspline' : VM : 'areaspline'
OUTPUT.AT<3> = "2013" : VM : "2012" : VM : "2011" : VM : "2010"
```

Area, stacked Chart



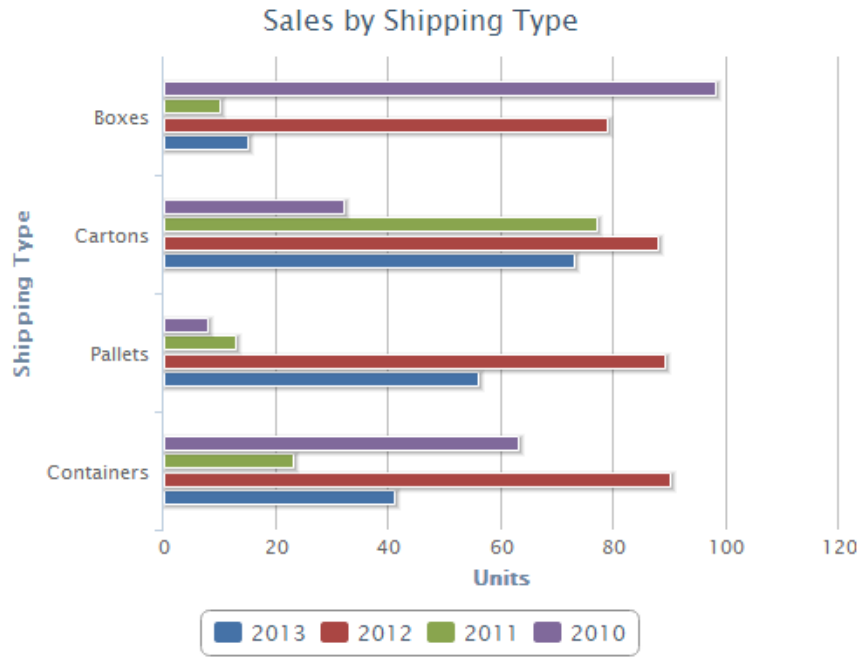

```
OUTPUT.AT<2> = 'area,stacked' : VM : 'area' : VM : 'area' : VM : 'area'
OUTPUT.AT<3> = "2013" : VM : "2012" : VM : "2011" : VM : "2010"
```

Area, stacked, percent Chart




```
OUTPUT.AT<2> = 'area,stacked,percent' : VM : 'area' : VM : 'area' : VM : 'area'
OUTPUT.AT<3> = "2013" : VM : "2012" : VM : "2011" : VM : "2010"
```

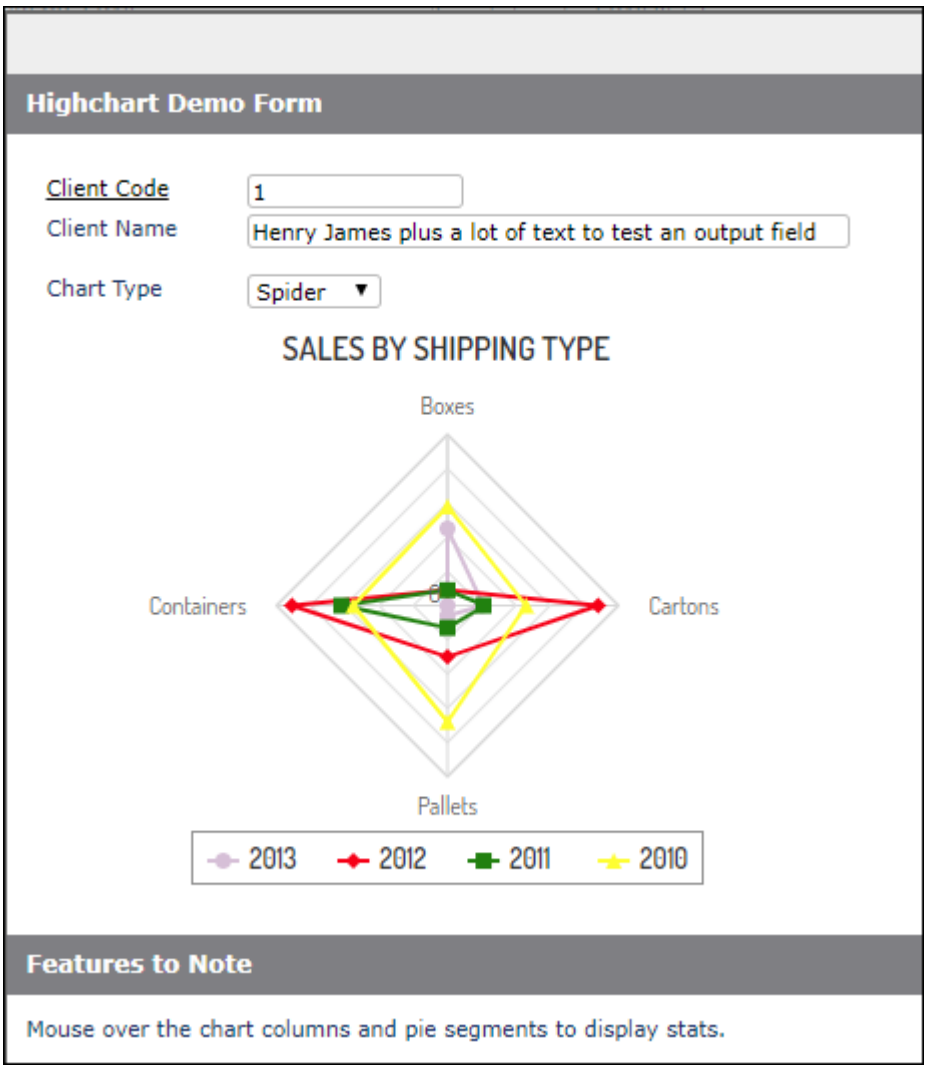
Bar Chart




```
OUTPUT.AT<2> = 'bar' : VM : 'bar' : VM : 'bar' : VM : 'bar'
OUTPUT.AT<3> = "2013" : VM : "2012" : VM : "2011" : VM : "2010"
```

Changes for Spider Charts

OUTPUT.AT<2> = 'line' : VM : 'line' : VM : 'line' : VM : 'line'
OUTPUT.AT<21> = ""
OUTPUT.AT<30> = ""



1) Chart node options
OUTPUT.AT<41> = "polar: true"

2) Plot options
OUTPUT.AT<42>

3) Series options
OUTPUT.AT<43> = "pointPlacement: 'on'"

4) Tooltip
OUTPUT.AT<44>

Two new attributes have been added for Spider Charts:

5) OUTPUT.AT<45> = node names – MV

6) OUTPUT.AT<46> = option string – MV by node (options to add to a node)

Example:

Add 2 options to xAxis:

OUTPUT.AT<45,-1> = 'xAxis'

OUTPUT.AT<46,-1> = "tickmarkPlacement: 'on', lineWidth: 0"

Add 3 options to yAxis:

OUTPUT.AT<45,-1> = 'yAxis'

OUTPUT.AT<46,-1> = "gridLineInterpolation: 'polygon', lineWidth: 0, min: 0"

The gridLineInterpolation: 'polygon' turns the graph from a “target” to be a “web”.
The nodes that can be utilised for your own options are:

- Title
- Subtitle
- xAxis
- yAxis

No Data to Display Message in Highcharts

Developers can add the following script to either Global or System Meta Data to get a default "No data to display" message:

```
<script src="charts/modules/no-data-to-display.js?v=4302" type="text/javascript"></script>
```

Click Events in Highcharts

The Highchart control is clickable. This provides the ability to have drill-into charting.

PROCESS.EVENT = “HICHART”
PROCESS.EVENTSOURCE = “Control Name”

DBVALUE = The value of the cell clicked

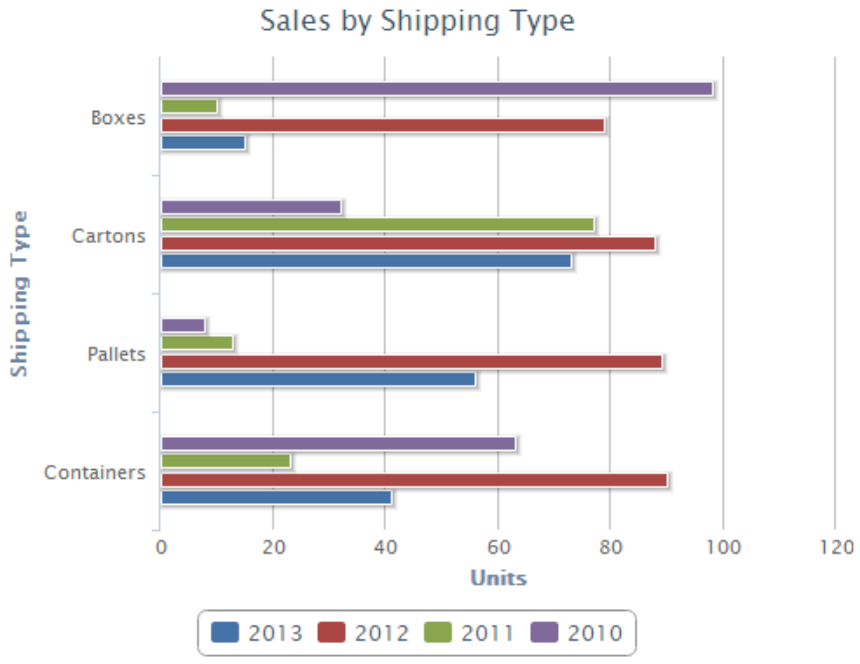
DBREPORT.CELL = The Series name *dot* column number

To implement you must add a Process After subroutine to the Hichart element. This will process the **HICHART** event that is triggered when the chart is clicked.

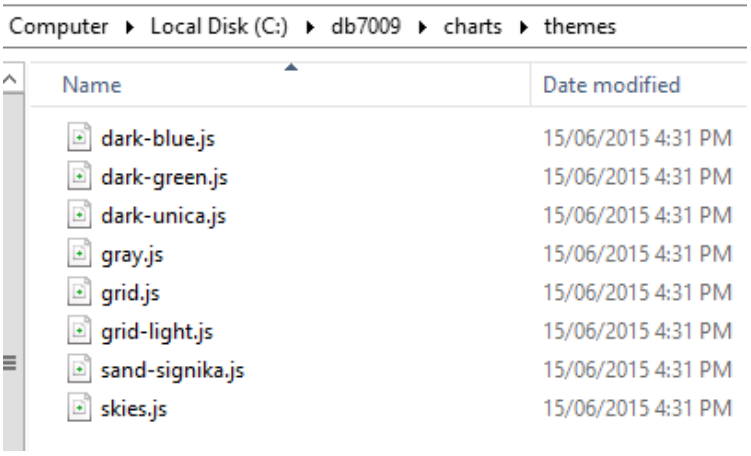
DBVALUE contains the data value for the clicked chart section.
DBREPORT.CELL contains the Row.Column clicked.

Changing the style of Highchart graphs

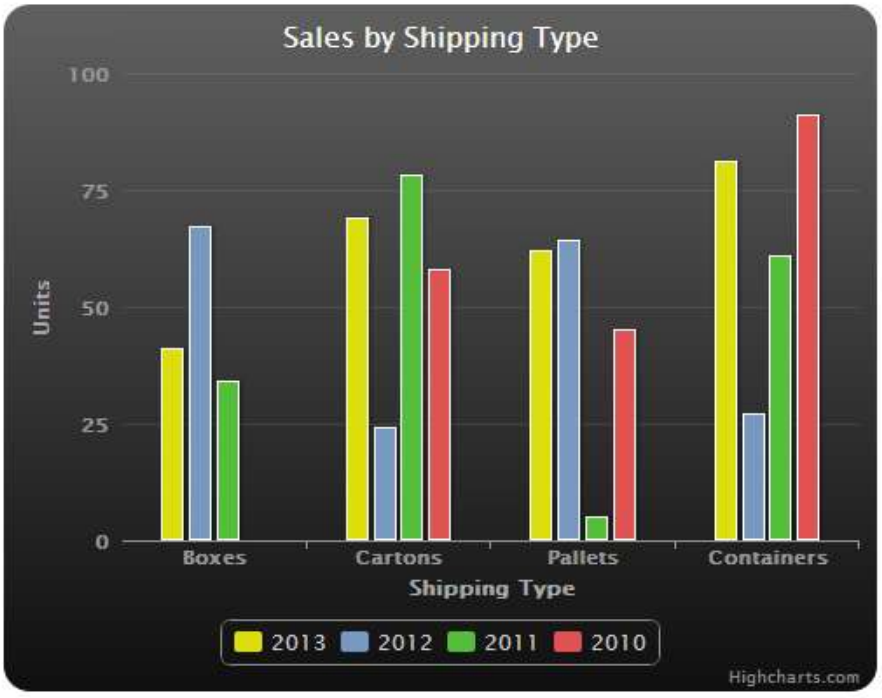
You can enter a theme to control the appearance of Highchart. By default, the theme is empty which produces charts like the following. Entry of a theme in Global or System Parameters will override the default.



The following themes are available:



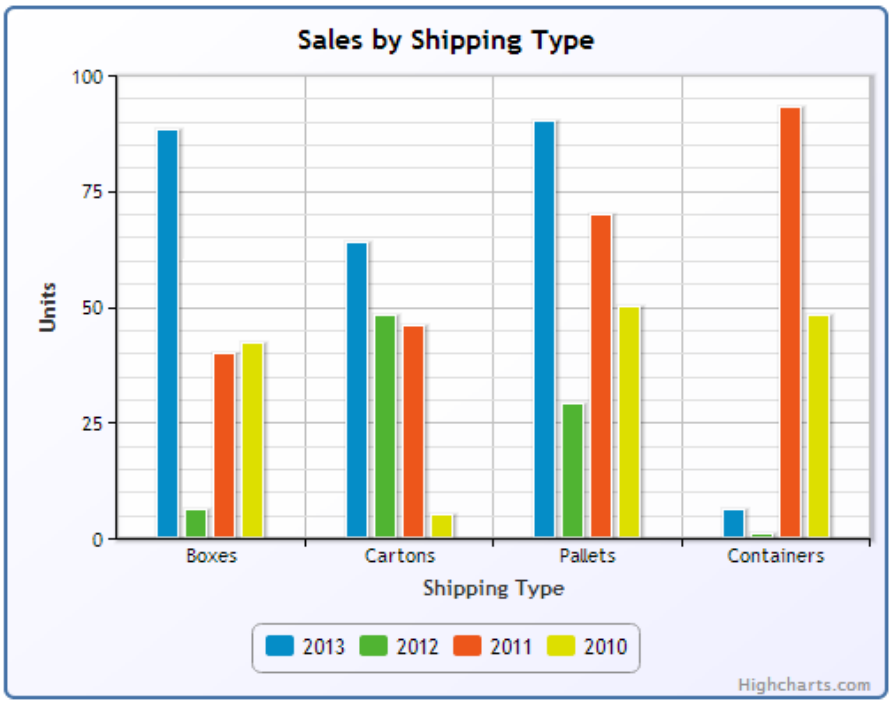
gray



dark-blue

dark-green

grid



skies

Themes can be modified or copied to a new name. In the example below, a new theme of demo.js was copied from the skies.js theme. This is stored in the charts/themes directory under the website.

In this example, the theme was modified to change the background image to:

plotBackgroundImage: 'images/ibaistest.jpg'

A border was also added.by **borderWidth: 1**

Charts Using Arrays in OUTPUT.REPORT

Column data in OUTPUT.REPORT(nn) may now contain an array as per:

3. An array of objects with named values. The following snippet shows only a few settings, see the complete options set below. If the total number of data points exceeds the series' **turboThreshold**, this option is not available.

```
data: [{
    x: 1,
    y: 9,
    name: "Point2",
    color: "#00FF00"
}, {
    x: 1,
    y: 6,
    name: "Point1",
    color: "#FF00FF"
}]
```

Thus each column of a chart may be a different color rather than all columns adopting the series colour.

In basic code set as follows:

```
OUPUT.REPORTS(nn)<row,1> = '{x: 1, y: 9, name: "Point2", color: "#00FF00"}'
OUPUT.REPORTS(nn)<row,2> = '{x: 1, y: 6, name: "Point1",  color: "#FF00FF"}'
```

Setting Values for Selected Chart Properties

Use OUTPUT.ATTR attributes 45 and 46 for defining multiple properties for specified options.

The OUTPUT.AT<45> can be multivalued:

```
NODE.OPTS = OCONV(THIS.ATTR<45>,"MCL")
NODE.VALS = THIS.ATTR<46>
```

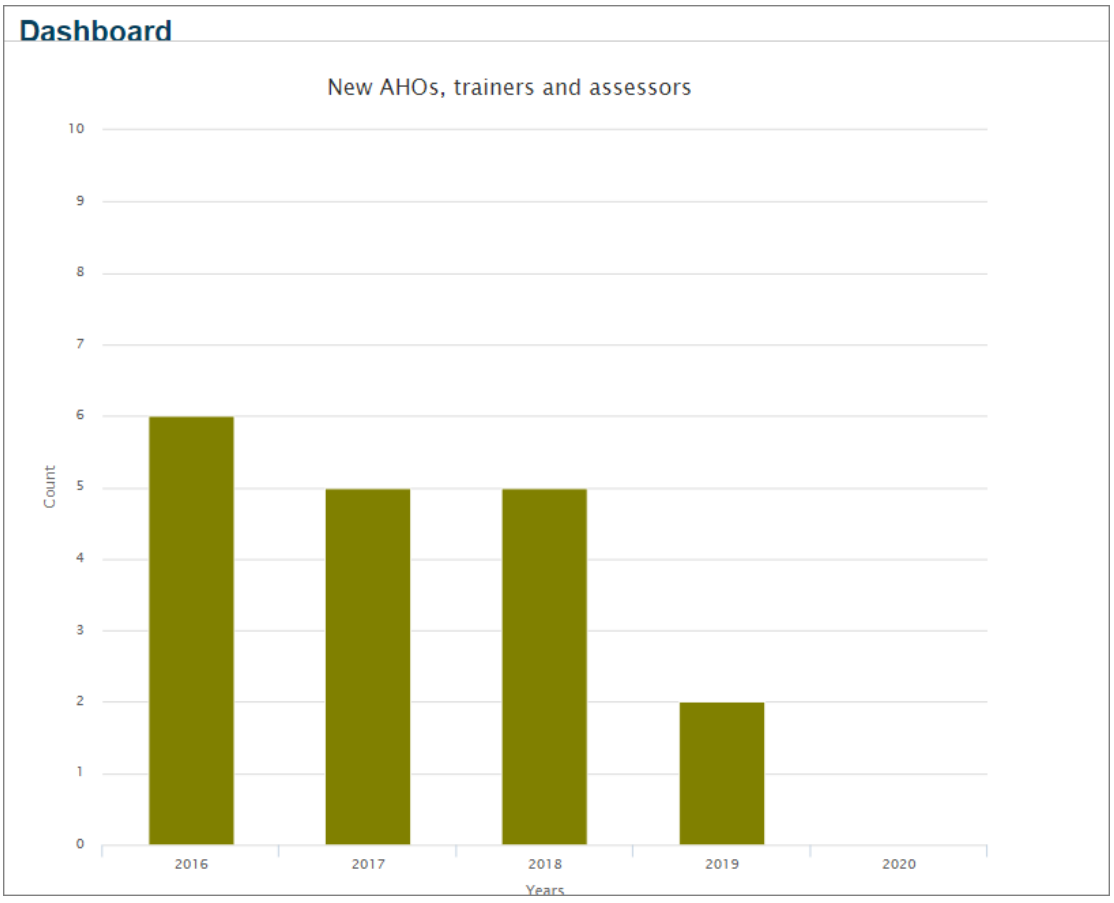
For example NODE.OPTS can be used to specify properties for things like:

- Title
- Subtitle
- Xaxis
- Yaxis

NODE.VALS is a comma separated list of HighChart API options for the associated NODE.OPTS.
Developers can reference the website https://api.highcharts.com/highcharts to find more details.

Here is an example of setting properties for the y-axis. Note that the property name is separated from the value by a colon. Here we define that Y-axis values will be integers and the scale will extend to 10. If the “max:10” property is not specified then the maximum value will be determined by the data.

```
1830    OUTPUT.AT<45,1> = "yaxis"
1831    OUTPUT.AT<46,1> = "allowDecimals:false,max:10"
```



Notes on the Highcharts example in Demonstration Account

The Highchart Sales Comparison Demo Form DBDEMO_CHART in the Demonstration Account provides a guide to assist the developer to implement a Highcharts chart. See this link for further details: http://api.highcharts.com/highcharts

Highcharts allows you to have a series of graphs plotted against each other. The Demo Form has only one graph and in order to get multiple colors applied to the columns it is necessary to use OUTPUT.ATTR<39> = 1. This causes the colorByPoint: true property to be added to the plotOptions Series array, which causes the colors that are defined for the legend to be used for the graph columns too. If the colors to be used are not listed then defaults will be applied by Highcharts.

Note that to create a background image the image name is passed as OUTPUT.AT<40> = 'images/db/dbSplash.jpg'.

Further, because there is only one graph, rather than a series, the values are assigned to each attribute of OUTPUT.REPORT like this:

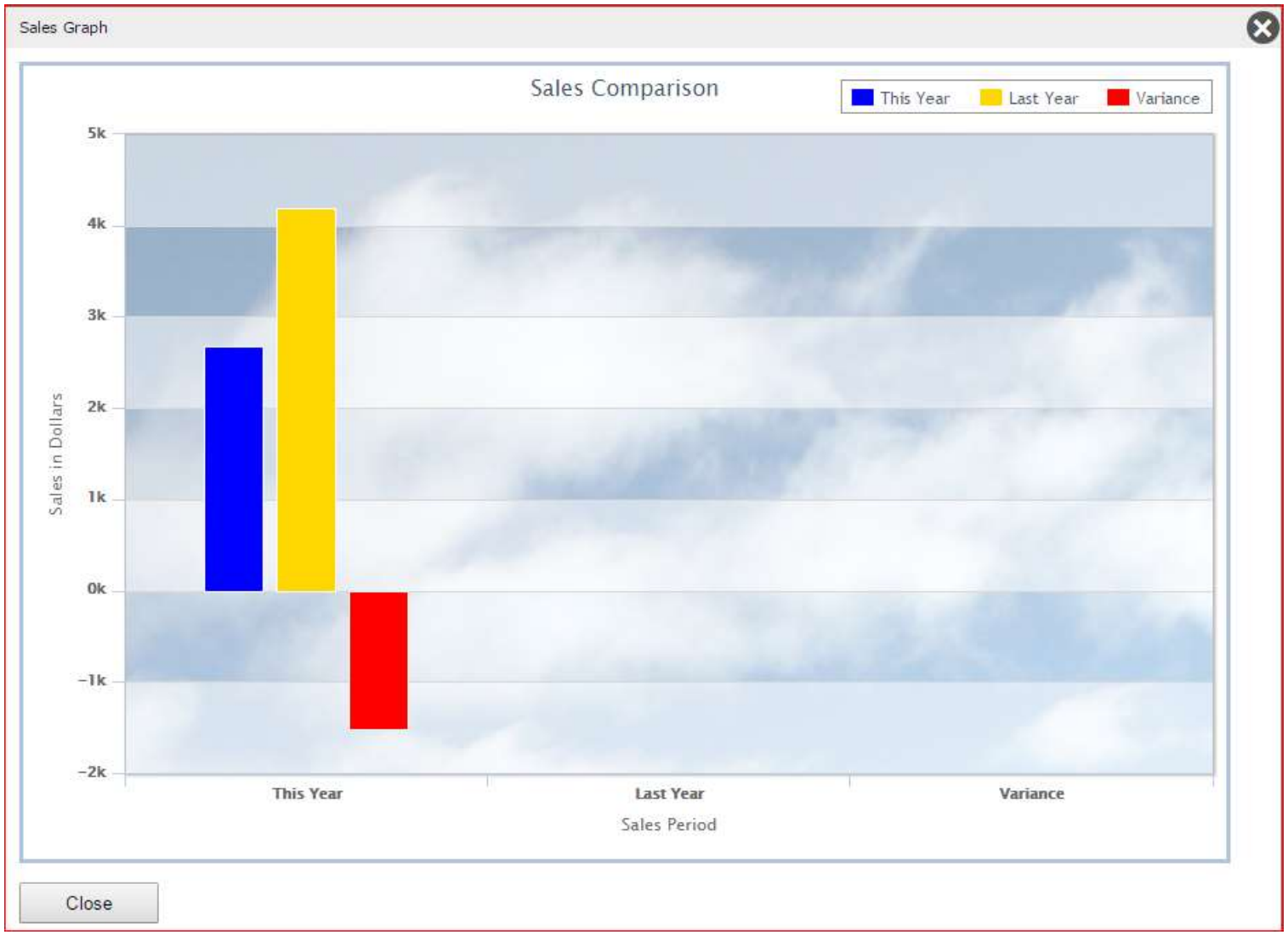
```
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<1> = OCONV(DBOTHER.RECORD(2)<DBS.YTD.SALES>,"MD2,")
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<2> = OCONV(DBOTHER.RECORD(2)<DBS.LY.SALES>,"MD2,")
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<3> = OCONV(DIFF,"MD2,")
```

The XML looks like this:

```
chart1 = new Highcharts.Chart({ chart: { renderTo: 'hichart1v3', type: 'column' },title: { text: 'Sales Comparison'}, colors: ['blue','gold','red'], xAxis: {categories: ['This Year','Last Year','Variance'], title: { text: 'Sales in Period'} }, yAxis: { title: { text: 'Sales in Dollars' } },plotOptions: {series: {colorByPoint: true }},series: [{ name: 'This Year', data: [2671.74,4188.65,-1516.91]}],});
```

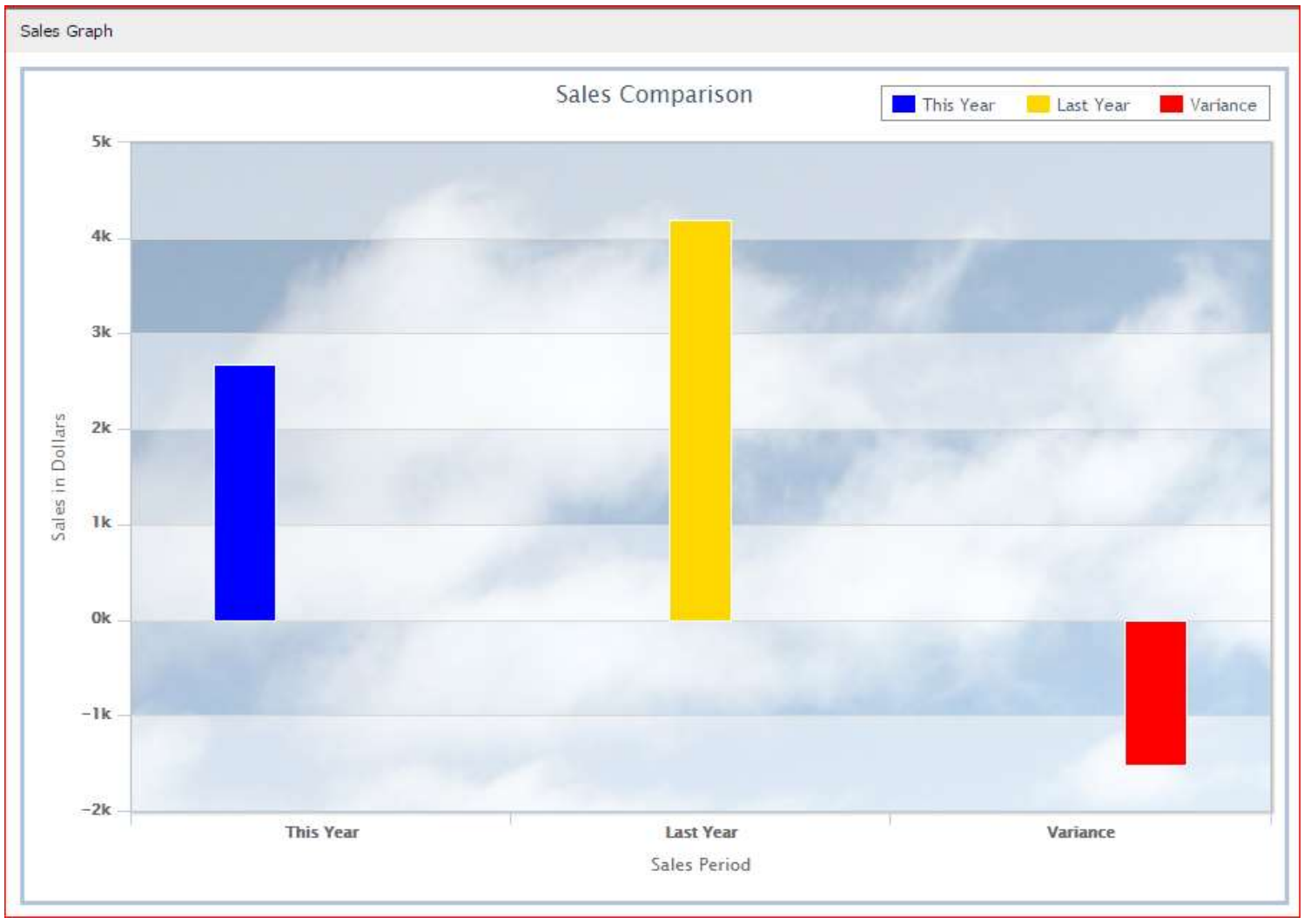
By way of clarification if the output rows are set up as values rather than attributes then the chart will render as shown below.

```
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<1,1> = OCONV(DBOTHER.RECORD(2)<DBS.YTD.SALES>,"MD2,")
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<1,2> = OCONV(DBOTHER.RECORD(2)<DBS.LY.SALES>,"MD2,")
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<1,3> = OCONV(DIFF,"MD2,")
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<2,1> = '0'
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<2,2> = '0'
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<2,3> = '0'
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<3,1> = '0'
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<3,2> = '0'
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<3,3> = '0'
```



Furthermore setting the output rows up as shown below will render the chart differently again.

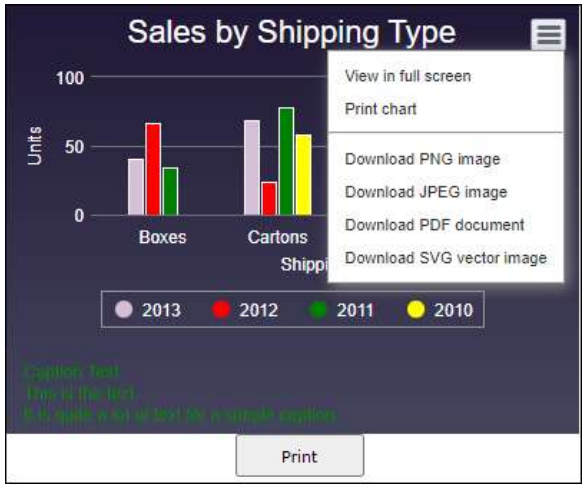
```
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<1,1> = OCONV(DBOTHER.RECORD(2)<DBS.YTD.SALES>,"MD2,")
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<1,2> = '0'
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<1,3> = '0'

OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<2,1> = '0'
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<2,2> = OCONV(DBOTHER.RECORD(2)<DBS.LY.SALES>,"MD2,")
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<2,3> = '0'

OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<3,1> = '0'
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<3,2> = '0'
OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<3,3> = OCONV(DIFF,"MD2,")
```



Note that the Blue column is on the left of 3 columns above “This Year”. The Yellow column is in the middle of the 3 columns above “Last Year” while the Red column is on the right of the 3 columns above “Variance”.

Highchart Context Menu

Click the hamburger icon to display the context menu.



Saving the Highchart as an image

The process to save the Highchart as an svg (Scalable Vector Graphics File) image is:

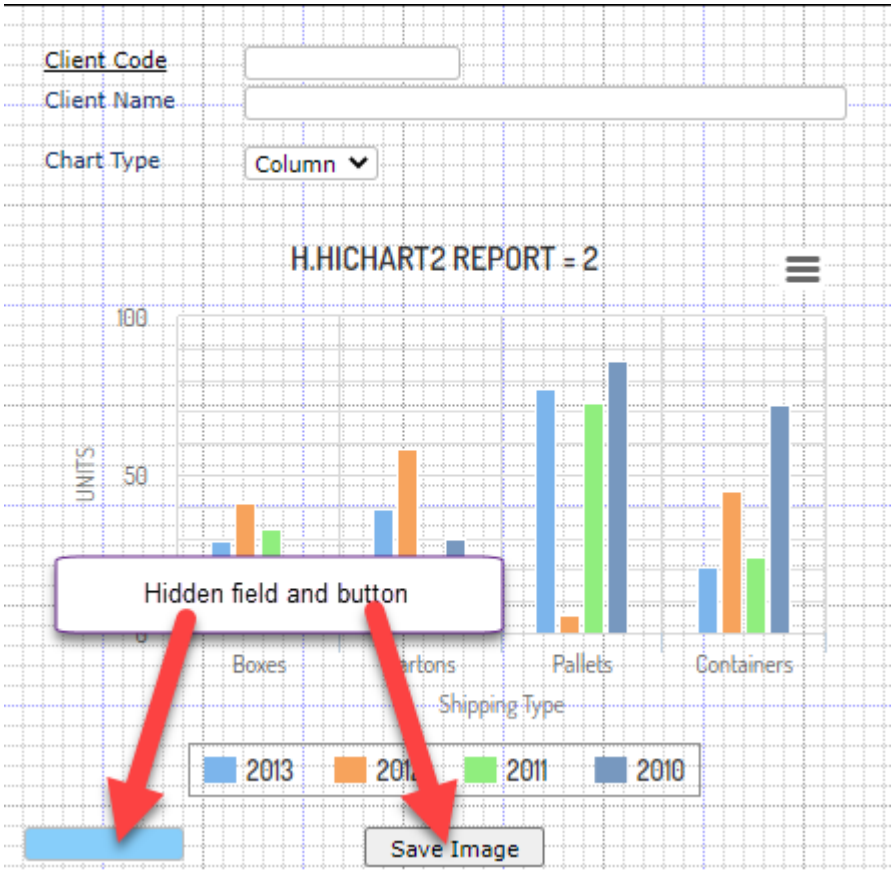
- Add a hidden input field(s) to the form containing the graph
- Set OUTPUT.AT<47> to the field name

```
*
BUILD.HICHART:
*
      PROCESS.REPORT.NUMBER = 2
      OUTPUT.REPORT(PROCESS.REPORT.NUMBER) = ""    ; * Variable used to store output in a report format
      OUTPUT.KEYS(PROCESS.REPORT.NUMBER) = ""      ; * Variable used to store key-links to OUTPUT.REPORT
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)= ""    ; * Variable used to store report headers
      OUTPUT.ATTR(PROCESS.REPORT.NUMBER) = ""      ; * Variable used to control report attributes
      OUTPUT.WIDTHS(PROCESS.REPORT.NUMBER) = ""    ; * Variable to store widths
      OUTPUT.ANCHOR(PROCESS.REPORT.NUMBER) = 0
      OUTPUT.LINE = ''
      OUTPUT.AT = ''
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<1> = "Boxes"
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<2> = "Cartons"
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<3> = "Pallets"
      OUTPUT.HEADERS(PROCESS.REPORT.NUMBER)<4> = "Containers"
      *
      OUTPUT.AT = "Sales by Shipping Type"
      IF DBWORK<DEM.HICHART.TYPE.WK> = "line" OR DBWORK<DEM.HICHART.TYPE.WK> = "spider" THEN
         OUTPUT.AT<2> = 'line' : VM : 'line' : VM : 'line' : VM : 'line'
      END ELSE
         OUTPUT.AT<2> = 'column' : VM : 'column' : VM : 'column' : VM : 'column'
      END
      OUTPUT.AT<3> = "2013" : VM : "2012" : VM : "2011" : VM : "2010"
      OUTPUT.AT<4> = 'thistle':VM:'red':VM:'green':VM:'yellow'
      *
      * Legend
      *
      OUTPUT.AT<12> = 1                  ; * Has Legend [Blank True, 0 No Legend]
      OUTPUT.AT<16,1> = 'bottom'         ; * [bottom] / top / left / right
      OUTPUT.AT<16,2> = 'horizontal'     ; * [horizontal] / vertical
      OUTPUT.AT<16,3> = 'center'         ; * [center] / right / left
      OUTPUT.AT<16,4> = 1                ; * borderwidth in pixels
      *
      * BackGround COlor
      *
      OUTPUT.AT<21> = "Shipping Type"        ; * X Axis Title [Blank No X Axis Title]
      *
      OUTPUT.AT<30> = "Units"  ; * Y Axis Title [Blank No Y Axis Title]
      *
      OUTPUT.AT<44> = "This is hover text for the chart title.\nTwo lines.\nThird line"
      *
      IF DBWORK<DEM.HICHART.TYPE.WK> = "spider" THEN
         * No axis labels
         OUTPUT.AT<21> = ""
         OUTPUT.AT<30> = ""
         * Chart option
         OUTPUT.AT<41> = "polar: true"
         OUTPUT.AT<43> = "pointPlacement: 'on'"
         * Other node options
         OUTPUT.AT<45,-1> = 'xAxis'
         OUTPUT.AT<46,-1> = "tickmarkPlacement: 'on', lineWidth: 0"
         OUTPUT.AT<45,-1> = 'yAxis'
         OUTPUT.AT<46,-1> = "gridLineInterpolation: 'polygon', lineWidth: 0, min: 0"
      END
      *
      * Optional field to save svg string
      *
      OUTPUT.AT<47> = 'DEM.HICHART.SVG1.WK'
      *
```

```
OUTPUT.ATTR(PROCESS.REPORT.NUMBER) = OUTPUT.AT
*
* LL loops through Boxes, Cartons, Pallets & Containers - xAXis
*
FOR LL = 1 TO 4
   *
   * RR Loops through the Years - multiple graphs
   *
   RRMAX=4
   FOR RR = 1 TO RRMAX
      YVAL = RND(100)
      OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<LL,RR> = YVAL        ; * Provide Data
      *
      * Data can be an array - see www.hicharts.com API documentation
      *
      * IF LL=2 THEN NEW.COLOR=",color:'black'" ELSE NEW.COLOR=''
      * OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<LL,RR> = "{y:":YVAL:NEW.COLOR:", name:'":OUTPUT.AT<3,RR>:"',}"

   NEXT RR
NEXT LL
*
PROCESS.TYPE<1,-1> = "H"
PROCESS.REFRESH<1,-1> = "H.HICHART2"
RETURN
```

The DesignBais subroutine DBI.G.DETAILNET will add the javascript to populate the hidden field with the svg. It is a complex XML string. You can inspect the string within the browser when running the DBDEMO*HICHART form if you are interested.

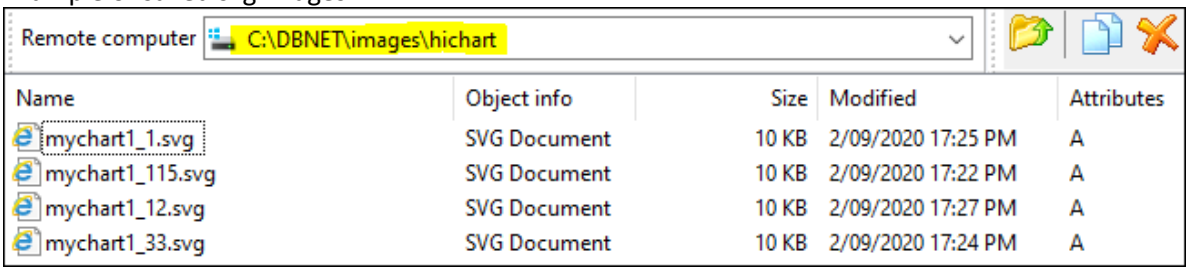
DesignBais then uses DBI.G.SAVEFILE to save the content in, for example, the website images folder after a button click or other event. In the example below the user clicks the *Save Image* button. The button event code is shown below.



```
*
BUTTON:
*
      BEGIN CASE
         CASE SCREEN.NO = "HICHART" AND EVENTSOURCE[1,10] = "B.SAVE.IMG"
            DBWORK<DEM.CLIENT.CODE.WK> = DBKEY
            IF EVENTSOURCE[LEN(EVENTSOURCE),1] = "1" THEN
               SF.PATH = 'images/hichart/mychart1_':DBWORK<DEM.CLIENT.CODE.WK>:'.svg'
               SF.CONTENT = DBWORK<DEM.HICHART.SVG1.WK>
            END ELSE
               SF.PATH = 'images/hichart/mychart2_':DBWORK<DEM.CLIENT.CODE.WK>:'.svg'
               SF.CONTENT = DBWORK<DEM.HICHART.SVG2.WK>
            END
            IF SF.CONTENT = '' THEN RETURN
            CALL DBI.G.SAVEFILE(SF.PATH,SF.CONTENT)
            PROCESS.STACK = "DBDEMO_RUN.SVG"
```

The above code allows for two hidden input fields. Only one field is implemented in the DBDEMO_HICHART demo form.

Example of saved svg images:



Example of selection process used in the DBDEMO_HICHART demo form report:

Example of highchart report showing spider graph:

Combining multiple report forms

DesignBais provides the ability to combine multiple report forms at runtime.

Typically this assignment would be performed in the subroutine assigned to the “Process Before Report” slot, which resides on the Report Generator definition form.

DBREPORT.INCLUDE

This variable provides the developer the ability to add extra report forms at run-time. Multiple reports can be included and are delimited by a value mark. A header section is mandatory on all reports even if the header section contains no fields. That is the report using DBREPORT.INCLUDE to include other reports at run time, and all the included reports.

Usage: DBREPORT.INCLUDE = “*Filename_Reportname[:VM:Filename_Reportname]* “

Eg.

```
DBREPORT.INCLUDE = “DBINVOICE_FOLLOWUPLETTER”:VM:”DBINVOICE_TERMSANDCONDITIONS”
```

In the above example, two more report forms are added to the original report form run. For this example let’s assume that the first report was “DBINVOICE_INVOICE”

The result is that three pages would print for every record selected in the report generation, being:

1.	DBINVOICE_INVOICE	Invoice printed first
2.	DBINVOICE_FOLLOWUPLETTER	Follow-up letter printed next
3.	DBINVOICE_TERMISANDCONDITIONS	Terms and conditions as the final page.

Note that the report selection process of the first report (the “first report” is the report that has the “Process Before Report” slot populated) controls the selection for all reports. The “Selection Name” process is mandatory for all reports but for included reports this selection is ignored. The “Process Before Report” and the “Report Read Process” are also ignored for included reports.

In general all reads for the reports that are combined using DBREPORT.INCLUDE must be done by the first report, since Included Reports are merged onto the end of the first report to become one larger report.

If an *included* report uses the same *Read Group* as the main report or another *included* report then these reads will clash and the report will not work correctly.

Similarly reads on fields with the same name on more than one report will not work.

If you have to place reads on subsequent reports then you must ensure that the above two conditions are obeyed. Placing all reads on the starting report will ensure that the report works correctly.

DBREPORT.EXCLUDE

It may be necessary to exclude pages in an include list (provided by DBREPORT.INCLUDE) for various records selected in the report generation. Multiple excludes are delimited by a value mark.

This assignment can only be completed in the “REPORT READ” event. This event is triggered if there is a subroutine named in the “Process After Read” slot in the report designer definition. The subroutine to control this is the one named in the first report in the “Report Read Process” slot.

Usage: DBREPORT.EXCLUDE = “*Filename_Reportname[:VM:Filename_Reportname]* “

Eg. Under certain conditions (using the example above) the follow-up letter is not required.

```
IF DBRECORD<MY.CONDITION> = TRUE THEN
   DBREPORT.EXCLUDE = “DBINVOICE_FOLLOWUPLETTER”
END
```

In this example only the invoice page and the terms and conditions page would print. The follow-up letter would be excluded for that record.

DBREPORT.INCLUDE.PAGE.BREAK

For each of the reports included in DBREPORT.INCLUDE a page break can be included before each page of the included set.

There are two possible values for this parameter.

Y Perform a page break before the included report

N Do not perform a page break before the included page.

Eg. Include a page break (using the example above) before each new page defined in DBREPORT.INCLUDE.
DBREPORT.INCLUDE.PAGE.BREAK = “Y”:VM:”Y”

Running Multiple Reports

In the “BEFORE REPORT” event of a report, you may set the variable **DBREPORTLIST** to a list of additional reports that are to be invoked. These additional reports will be appended to the first report in the printed output or in the preview window.

Usage:

DBREPORTLIST = *myfile_myreport*: VM : *myfile_myreport*

Change Account and Run Specified Form

This section describes a method of providing functionality to change account and run a different form to the user's normal start form in the target account.

There is a DesignBais Demo Form that demonstrates this feature. See form DBDEMO_LOGTO.

The method can be used to run a base form, not a layered or modal form.

Code similar to that shown below is used to initiate the change of account. Note that the name of the target account must be in DBVALS<1,1> and the name of the form to run must be in DBVALS<1,2>. DBVALS is passed into DBI.G.DA as the second argument. The *action* must be set to "U" which is passed in as the first argument.

Other data may be passed in via DBVALS, such as a key or other inputs to the target form. See the AFTER DISPLAY code segment below.

DBVALS<1,11> and DBVALS<1,12> must contain the account name and form that are to run when the the target process is complete and control returns to the starting account. The process must return to the same account which is held in the common variable DBIACCOUNT. (DBIACCOUNT contains the name of the account that the user is in at the time of clicking the *B.LOGTO* button). The form DBDEMO_DEMO will run on return.

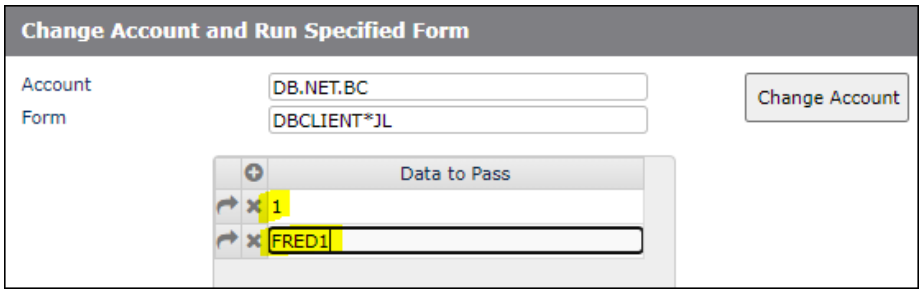
DBVALS<1,11> is set to the value of DBIACCOUNT indicating that the process will return to this account.

When DBI.G.DA finds a match between DBVALS<1,1> and DBVALS<1,11> with DBACT set to "U" then it sets DBIACCOUNT to the target account name, thus triggering the change of account.

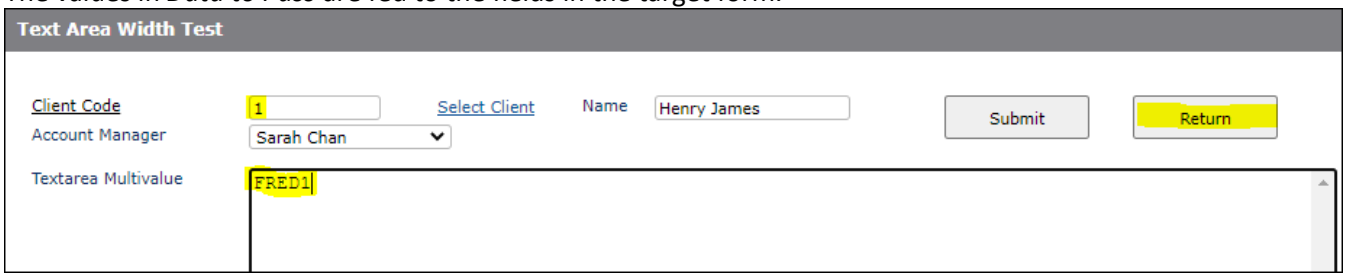
```
CASE SCREEN.NO = "LOGTO" AND EVENTSOURCE = "B.LOGTO"
   * Must be in a base form
   IF DBWLEVEL > 2 THEN
      IERR.TEXT='Please run from a base form.'
      CALL DBI.G.GLOSSARY(IERR.TEXT)
      RETURN
   END
   DBWORK<DEM.WORK2.WK> = CHANGE(DBWORK<DEM.WORK2.WK>,'*','_')
   * set target Account and Form - must be in position 1 & 2 respectively
   DBACT = 'U'
   DBVALS = ''
   DBVALS<1,1> = DBWORK<DEM.WORK1.WK>  ;* DBIACCOUNT is set if user has access to the target account
   DBVALS<1,2> = DBWORK<DEM.WORK2.WK>  ;* for this example the form name is "JL"
   * Add data to pass in MV 3 to 10
   FOR J=1 TO 8
      DBVALS<1,-1> = DBWORK<DEM.WORK19.WK,J>
   NEXT J
   * set the return stack variables
   * In this case return to the current account and the Demo Form
   DBVALS<1,11> = DBIACCOUNT
   DBVALS<1,12> = "DBDEMO_DEMO"
   * Add extra data to return in MV 13 to 20 if required
   CALL DBI.G.DA(DBACT,DBVALS)
```

The *AFTER DISPLAY* event in the target form uses the *action "R"* to return the values passed in DBVALS.
The subroutine DBI.G.DA shuffles DBDEVACTIONS<1,11> through <1,20> back into DBDEVACTIONS<1,1> through <1,10>. So at the time of the AFTER DISPLAY in the target account the values in DBVALS<1,2> and DBVALS<1,3> are passed to the designated fields on the target form.

```
*
AFTER.DISPLAY:
*
  * Read in existing data before screen display:
  BEGIN CASE
     CASE FIELD(SCREEN.NO,"~",1) = "JL"
        DBACT = 'R'
        DBVALS = '';* will look for SCREENROOT
        CALL DBI.G.DA(DBACT,DBVALS)
        IF DBVALS # '' THEN
           DBPASS.DBVALUE.TO = 'DBC.CLIENT.CODE'
           DBPASS.DBVALUE = DBVALS<1,2>
        END
        IF DBVALS<1,3> # '' THEN
           DBPASS.DBVALUE.TO<1,-1> = 'DBC.TEXTAREA.MV'
           DBPASS.DBVALUE<1,-1> = DBVALS<1,3>
        END
```



The values in *Data to Pass* are fed to the fields in the target form.



A button on the target form is used to set DBIACCOUNT to the name of the original account and to stack, using PROCESS.STACK, the name of the form that is to run when control comes back to the original account.

```
*
BUTTON:
*
      BEGIN CASE
         CASE EVENTSOURCE = "B.RETURN" AND FIELD(SCREEN.NO,"~",1) = "JL"
            DBACT = 'S'
            DBVALS = ''
            CALL DBI.G.DA(DBACT,DBVALS)
```

The developer must provide a means for the user to return to the calling account and form from the form in the target account. In the above demo form this was achieved by a button. In practice this can be triggered by a submit or clear button, or by a specific return button on the form.

The use of the value positions of DBDEVACTIONS is documented below.

DBI.G.DA(DBACT,DBVALS)

```
* DBDEVACTIONS is a single attribute with MVs & SVs.
*
* Existing usage will be maintained for MVs 1 to 5 as per:
*  DBDEVACTIONS<1,1,1>    = File for DBIFORMS*D10
*  DBDEVACTIONS<1,1,2>    = Form for DBIFORMS*D10
*
*  DBDEVACTIONS<1,2,1>    = File for DBIREPORTS*D10
*  DBDEVACTIONS<1,2,2>    = Report for DBIREPORTS*D10
*
*  DBDEVACTIONS<1,3,1>    = File for DBISELECT*D10
*  DBDEVACTIONS<1,3,2>    = Selection for DBISELECT*D10
*
*  DBDEVACTIONS<1,4,1>    = File for DBIFILES*D10
*
*  DBDEVACTIONS<1,5,1>    = File for DBIPROP*D10
*
*  DBDEVACTIONS<1,11,-1>   = List of Target Forms
*  DBDEVACTIONS<1,12,-1>   = For the target forms above must be the File name
*  DBDEVACTIONS<1,13,-1>   = For the target forms above must be the Form, Report or Selection name
*  DBDEVACTIONS<1,14,-1>   = Application controlled
*  ...
*  DBDEVACTIONS<1,20,-1>   = Application controlled
*
*  DBDEVACTIONS<1,21,-1>   = List of Return Forms - associated with Target Form (where to go back to)
*  DBDEVACTIONS<1,22,-1>   = For the Return forms above must be the File name
*  DBDEVACTIONS<1,23,-1>   = For the Return forms above must be the Form, Report or Selection name
*  DBDEVACTIONS<1,24,-1>   = Application controlled
*  ...
*  DBDEVACTIONS<1,30,-1>   = Application controlled
*
*
* DBACT    = U - Update
*            = D - Delete
*            = R - Return Values
*            = S - Set PROCESS.STACK and move Return MVs
*            = A - Account change remove any other values
*
* DBVALS    = MV list of values to add to DBDEVACTIONS or to return.
*             First MV is the target form.
*             Next 9 MVs are for application use.
*             If target form is DBIFORMS*D10 than MV 2 must be the File & MV 3 must be the Form,
*             similarly for DBIREPORTS*D10, DBISELECT*D10, DBIFILES*D10 & DBIPROP*D10
```

Controlling Button Appearance

DesignBais allows for three button event states.

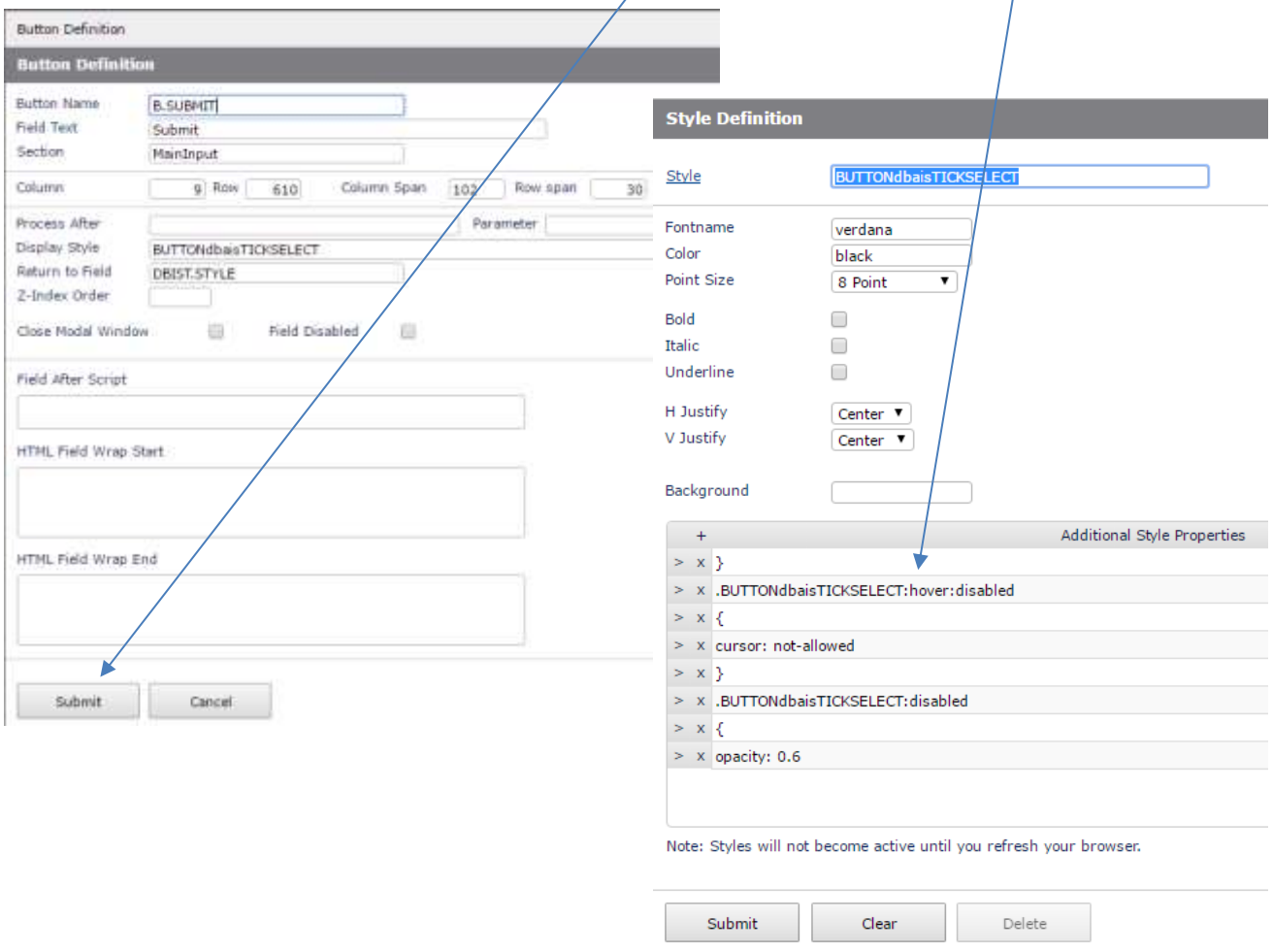
Each button state is defined by position in the Display Style separated by a comma.

Eg. BUTTONdbaisTICKSELECT , BUTTONdbaisHOVER, BUTTONdbaisLEAVE

If no value is entered in a comma position, no style change will apply.

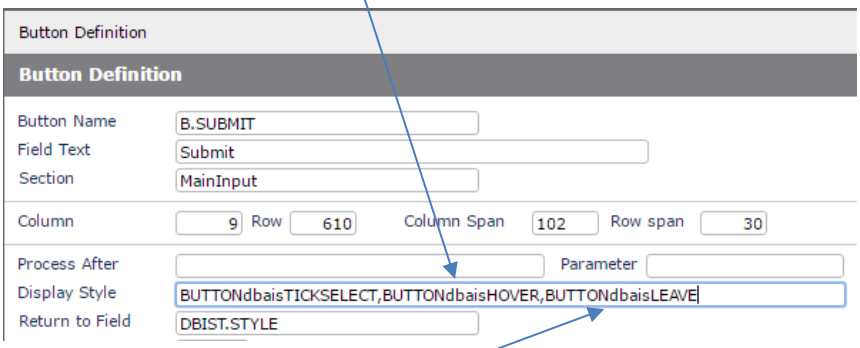
1. **Normal no focus (position 1)**

   This state is the normal state for the button. It is controlled by the Style assigned to the button. In the below example you can see the style that is being used for the button that appears as the “Submit” on the bottom on the form. Only the normal position is set to the style property. In this example the hover effect is obtained from the Additional Style Properties of the button style.



2. **Hover - focus (position 2)**

When the mouse is focused over a button, a hover effect can applied. This process essentially replaces the style of the button with another style when the hover is activated.



3. **Leave focus (position 3)**

If you wish to nominate a style to apply when the mouse focus is lost from a button, the third parameter can be used.

4. The styles “dbaisdevp1” to “dbaisdevp7” are no longer used in forms and report designer.

A new style called BUTTONdbaisDev has been created with sub classes for different hover colors. Refer to the buttons in the Forms Designer grid.

For example the Exit button class is: ”BUTTONdbaisDev BUTTONdbaisDevRed”

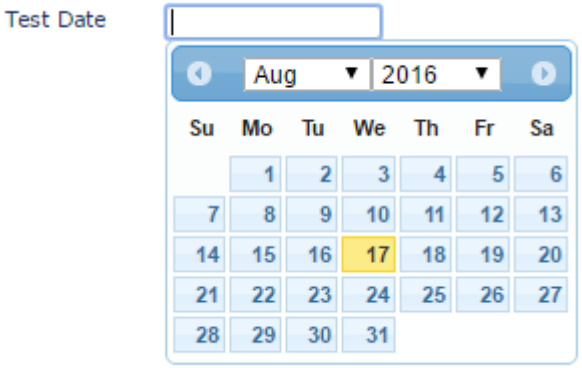
The “labelcontainer” images are now redundant as well.

DesignBais Subroutines and Standard Forms

DesignBais has a number of subroutines that can be called to perform various functions.

DBI.G.CALENDAR

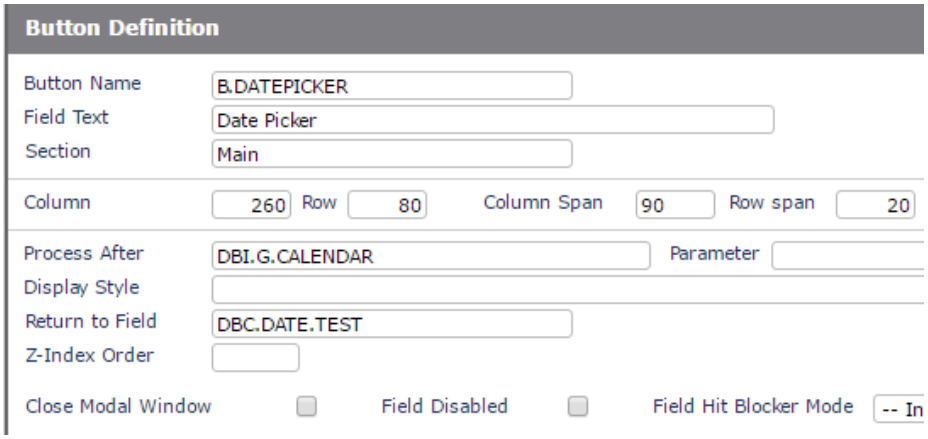
DesignBais can invoke a Date Picker Calendar display for all fields on a form that have a Field Type of DATE. By default the date picker will not display as the user tabs into or focuses on a field. This behaviour can be controlled by changing the **Show Popup Calendar** setting in the Global and System Parameters maintenance forms. DesignBais will check Global then System Parameters to determine whether to display the Date Picker on focus. If these are set to "Default" then the default behaviour is used.



To invoke the Date Picker from a button you must set up the Process After and the Return to Field:

The "Process After" (or PROCESS.STACK if invoked from a program,) name is **DBI.G.CALENDAR.**

The "Return to Field" for the button (or DBRETURN.TO.FIELD variable if invoked from a program) should be set to the name of the field that the date is to be returned to.



For releases prior to 7.0.0.0 DesignBais has a standard calendar form that you can invoke from any button.

The "Process After" (or PROCESS.STACK if invoked from a program,) name is **DBIPARMS_CALENDAR**

The "Return to Field" for the button (or DBRETURN.TO.FIELD variable if invoked from a program) should be set to the name of the field that the date is to be returned to.

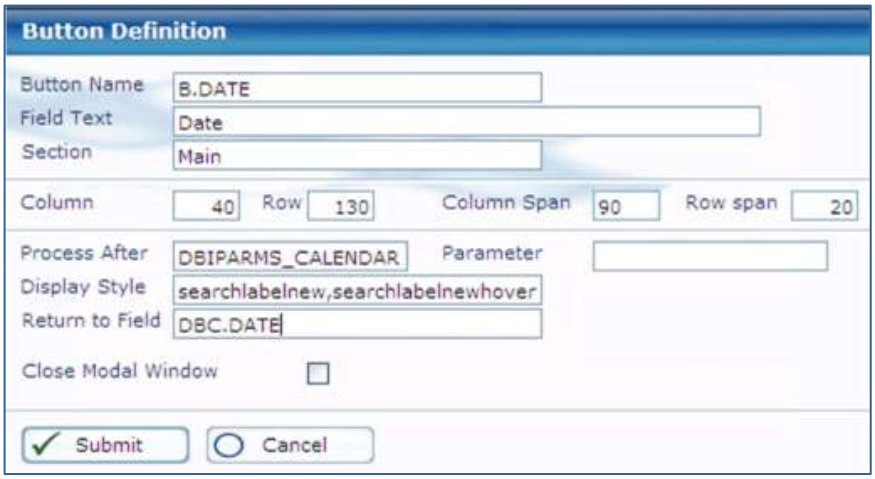
If the date field is in a multi-value grid then the following code can be used:

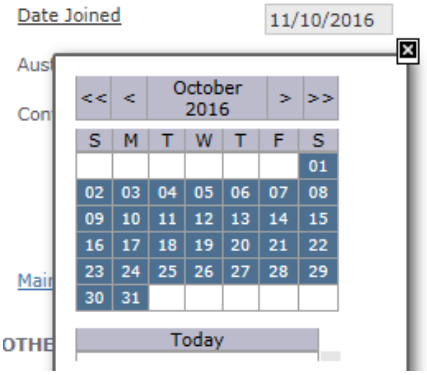
```
PROCESS.STACK = "DBIPARMS_CALENDAR"
DBRETURN.TO.FIELD = 'DBC.INV.DATE~|':DBMVCOUNT
```

where DBMVCOUNT corresponds to the row of the date field to be populated. This basic code can be linked to another column in the grid which has the '*Click event on process after'* enabled.

If the field named in the "Return to Field" prompt already has a date, the date in the calendar will default to the same.

Example Button Definition





The style group of the calendar form will match that of the calling form.

DBI.G.GLOSSARY

Move the error message to the glossary. This routine should be called from a BASIC subroutine whenever IERR.TEXT is assigned a value.

Usage: **DBI.G.GLOSSARY(*Message String)***

Example:
```
IERR.TEXT = "Unable to open the file [DBCLIENT]"
CALL DBI.G.GLOSSARY(IERR.TEXT)
```

The text (excluding the text between '[' and ']') will be added to the system glossary. This allows for the implementation of a system in another language without having to re-code the errors in subroutines. Attribute and Value marks will be replaced by space characters in Message String before the string is added to the glossary.

DBI.G.DIALOG

This subroutine is used to control the display of a message dialog.

Usage: **DBI.G.DIALOG(*DBBUTTON.CONFIG,DBDIALOG.TEXT,DBDIALOG.PROG,DBDIALOG.PARAMETER*)**

The current program calls to DBI.G.DIALOG only to set variables, not to launch a form. Control is immediately returned back to the current program and the next executable line.

On RETURN back to the DesignBais control programs, the defined dialog window will be presented to the user.

When the user makes a selection, the program named as the DBDIALOG.PROG is called. This is an application program that processes DesignBais events just like the current program making the call. The program specified could be the same event handling program that is making the call. This is not recursive because the current program has finished processing and has returned control to the user.

On execution of the subroutine named in DBDIALOG.PROG, the PROCESS.EVENT variable will be “DIALOG”. This is an event like any other. Rather than the name of a form field, the inbound PROCESS.EVENTSOURCE is the text passed to the DBDIALOG.PARAMETER. In essence the developer is passing a text message to explain what the dialog box was.

In the Dialog event handler, the developer checks this message, and then processes the response. As an example, if the message is “DELETE TRANSACTION”, then the developer knows what question the user answered, and the code can act accordingly. The user response comes with the DIALOG event in the PROCESS.PARAMETER variable.

In Release 7 and above the following applies:

Buttons
0 OK button only.
1 OK and Cancel.
2 Abort, Retry, and Ignore – not implemented, mapped to option 3.
3 Yes, No, and Cancel.
4 Yes and No.
5 Retry and Cancel – not implemented, mapped to option 1.

Icons and Default Buttons have been re-implemented from Release 7.5.1.1.

The four arguments associated with the DBI.G.DIALOG subroutine are defined as follows:

DBBUTTON.CONFIG

Buttons	Icon	Default Button
0 OK button only.	16 Critical Message (Stop sign)	0 First button is the default.
1 OK and Cancel	32 Warning Query (Question)	256 Second
2 Abort, Retry, and Ignore	48 Warning Message (Exclamation) (Deprecated)	512 Third
3 Yes, No, and Cancel	64 Information Message (**i**)	768 Fourth
4 Yes and No		
5 Retry and Cancel (Deprecated)		

Add the numbers to create the desired dialog configuration.

Example: 32 + 3 = 35

32 displays a question mark icon

3 displays YES NO and CANCEL buttons.

If we add 256 to this then the second button (NO) becomes the default button. That means if the users presses the ENTER key rather than clicking a button, the second button is treated as the default response.

If the users clicks **X** to close the dialog this action is always treated as a CANCEL (even when there is no CANCEL button is present on the dialog). Therefore, it is important to always check for this condition.

DBDIALOG.TEXT

The text to appear in the dialog window

DBDIALOG.PROG

The BASIC subroutine to call after the user clicks a dialog button

DBDIALOG.PARAMETER

The entry point within the BASIC subroutine to be used as a reference. This parameter updates the common variable PROCESS.EVENTSOURCE, which can be used within a BASIC subroutine.

The variable PROCESS.PARAMETER contains the value of the button that was pressed.

The responses assigned to PROCESS.PARAMETER are:

1 OK button was clicked.

2 Cancel button was clicked.

3 Abort button was clicked.

4 Retry button was clicked (pre-Release 7 only).

5 Ignore button was clicked (pre-Release 7 only).

6 Yes button was clicked.

7 No button was clicked.

The following is an example of handling for a Dialog event where the DBDIALOG.PARAMETER was set to “client on hold”.

```
* Other event handling code here, then...
          CASE PROCESS.EVENT = "DIALOG"
             GOSUB DIALOG
             RETURN
          ...
        RETURN

        DIALOG:
           BEGIN CASE
              CASE EVENTSOURCE = "client on hold"
                 BEGIN CASE
                    CASE THIS.PARAMETER = 6 ;* Yes Button
                       * Set the flag to put the client on hold
                    CASE THIS.PARAMETER = 7 ;* No Button
                       * Return back to the calling field
                       DBRETURN.TO.FIELD = "DBC.CLIENT.NAME"
                    CASE THIS.PARAMETER = 2 ; * Always handle!
                       * User clicked cancel or closed with X
                       * return to same field as if No was clicked
                       DBRETURN.TO.FIELD = "DBC.CLIENT.NAME"
                    CASE 1 ; * should never happen, handle anyway
                       * send message to developer, then...
                       DBRETURN.TO.FIELD = "DBC.CLIENT.NAME"
                 END CASE
              CASE PROCESS.EVENTSOURCE = "change client contact”
                 ...
           END CASE
        RETURN
```

DBI.G.SENDEMAIL

This subroutine is used to send an email from a DesignBais application.

Using email within DesignBais requires the db.config file to be set up. Refer to the Web Component Manual. In summary you must set the values in admin/db.config (the following shows example data – replace with your settings):

```
<smtpHost>192.168.199.1</smtpHost>
<smtpPort>25</smtpPort>
<smtpEnableSsl>false</smtpEnableSsl>
<smtpHTMLFormat>true</smtpHTMLFormat>
<smtpLogin></smtpLogin>
<smtpPassword></smtpPassword>
```

Note that if an external email server is used then that server should allow relaying of emails originating from the web server.

Usage: **DBI.G.SENDEMAIL(EMAIL.TO,EMAIL.FROM,EMAIL.CC,EMAIL.ATTACHMENT,EMAIL.SUBJECT,EMAIL.MESSAGE)**

There are six parameters associated with the call.

EMAIL.TO	The email address of the recipient. Multiple addresses can be separated by ";"
EMAIL.FROM	The email address of the sender.
EMAIL.FROM<2>	A valid email address but can be in the form "anyName anotherName <emailaddress>".
EMAIL.FROM<3>	The reply to email address.
EMAIL.CC	Email address for copies.
EMAIL.CC<2>	Email address for blind copies.
EMAIL.ATTACHMENT	The filename of any attachment.
EMAIL.SUBJECT	The subject of the email.
EMAIL.MESSAGE	The message content on the email. The message can be attribute mark delimited for multi-line.

DesignBais email functionality can be tested using the test page admin/emailTest.aspx.

The test program uses the email server parameters as specified in admin/db.config. These parameters are:

- smtpHost
- smtpPort
- smtpEnableSsl
- smtpHTMLFormat
- smtpLogin
- smtpPassword

An image can be included in the EMAIL.MESSAGE by using a link to a folder that is accessible to the web server. For example:

EMAIL.MESSAGE = 'Please see the attached image:'
EMAIL.MESSAGE:='
<img src="http://192.168.199.194/db7009/images/db/dbLogoGray.png">
'

Regarding the path for email attachments the DBI.G.SENDEMAIL routine simply encodes the email variables for javascript, XML & HTML and passes an XML packet to the web server:

```
OUT.STRING<-1> = '<action>'
OUT.STRING<-1> = '<name>sendMail</name>'
OUT.STRING<-1> = '<to>':EMAIL.TO:'</to>'
OUT.STRING<-1> = '<from>':EMAIL.FROM:'</from>'
OUT.STRING<-1> = '<cc>':EMAIL.CC:'</cc>'
OUT.STRING<-1> = '<attachment>':EMAIL.ATTACHMENT:'</attachment>'
OUT.STRING<-1> = '<subject>':EMAIL.SUBJECT:'</subject>'
OUT.STRING<-1> = '<message>':EMAIL.MESSAGE:'</message>'
OUT.STRING<-1> = '</action>'
```

Note that the attachment path must have a "/" or "\" character in it.

Consider a website located at *http://localhost/db* with the corresponding physical folder at *c:\db*.
If there is a file *c:\db\uploads\myfile.txt* then the virtual path becomes*: /db/uploads.myfile.txt*.

In basic code the developer can can set the email attachment path as either:

- a physical path such as *c:\db\uploads\myfile.txt*

or

- a virtual path such as */db/uploads/myfile.txt*

or

- indeed this path may work *"../uploads/myfile.txt*".

DBI.G.SENDSNS

This subroutine is used to send an SNS message from a DesignBais application.

Amazon Simple Notification Service (SNS) is a notification service provided as part of Amazon web services. It provides a low-cost infrastructure for the mass delivery of messages, predominantly to mobile users.

Usage: **DBI.G.SENDSNS(SNS.PARAMS,SNS.SUBJECT,SNS.MESSAGE,SNS.PHONE.LIST,RTN.PROGRAM, RTN.PARAM)**

There are six parameters associated with the call.

SNS.PARAMS	The SNS parameters as defined in the Amazon SNS form in either System or Global Parameters.
SNS.SUBJECT	The message heading.
SNS.MESSAGE	The message to be sent. Mandatory.
SNS.PHONE.LIST	The list of phone numbers to receive the message. Can be delimited by attribute, value or sub-value marks. At least 1 number must be specified.
RTN.PROGRAM	The name of the subroutine to be invoked on return from the web service call. Mandatory.
RTN.PARAM	The string to be used for the event source (PROCESS.EVENTSOURCE) in the subroutine that is invoked on return from the web service call.

DBI.G.PRINTFORM

Will print a copy of the current form including the data displayed on the form.

Can be called from a button or from a BASIC subroutine.

Parameters associated with the call are to be entered into the Parameter slot of a button or into PROCESS.PARAMETER if being called via a subroutine. Parameters are separated by a comma.

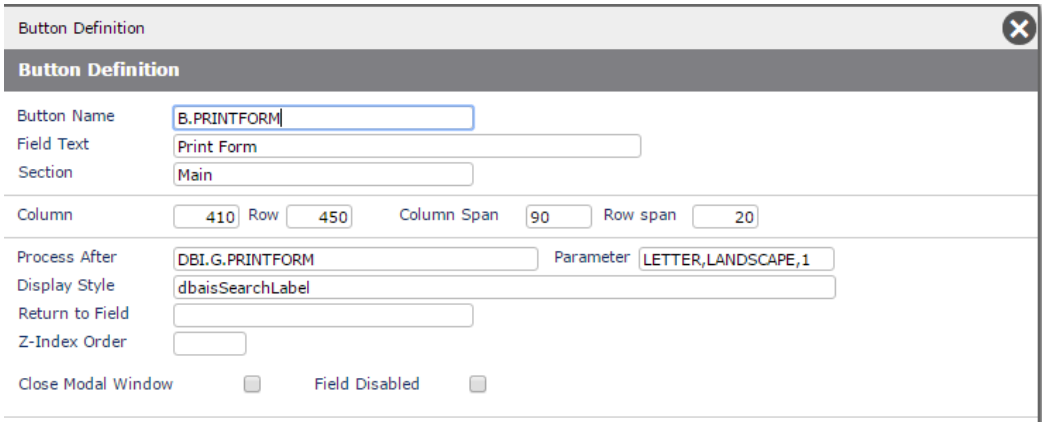
Parameters are:

Form type	Letter, Legal or A4
Form Orientation	LANDSCAPE or PORTRAIT
Preview Mode	1 for preview, 0 for direct to printer
Fields to Print	List of fields to print (separated by semi-colon)

Example 1 **From a subroutine call**

```
PROCESS.PARAMETER = “Letter,LANDSCAPE,1”
CALL DBI.G.PRINTFORM
```

Example 2 **From a button**



Currently the Convert to PDF option is not invoked.

Developers should note that certain functions may be interrupted by a call to DBI.G.PRINTFORM.

An example is DBSECTIONSPEC which is cleared by the DesignBais engine after acting on its value. So if DBSECTIONSPEC has been used to, for example, collapse a section and the user clicks a button to invoke DBI.G.PRINTFORM then within the print form routine DBSECTIONSPEC will be null. The printed view of the form will have a gap where the collapsed section would appear.

The developer should re-apply the DBSECTIONSPEC action in the MODAL RETURN event from DBIPARMS_M31. This will restore the form to the same state as it was prior to calling the print form routine.

Technical Note: DBFIELDSTATES holds the list of form fields not collapsed or hidden and is used to pick the fields to display in DBI.G.PRINTFORM. This means that collapsed and hidden fields are not included in the print form view.

In a future release of DesignBais the blank space on the printed form corresponding to a collapsed section will be removed by reference to DBCOLLAPSELIST held on the DBISESSIONS file.

DBI.G.SECURITY.RECORD

Is used to determine whether a record passes the DesignBais Entity Security model.

Typically, if the application is using DesignBais Entity Security, this routine would not be used. There are instances though where DesignBais may not be doing a read, but the developer wants the record read checked against the entity security model.

Usage: **SUBROUTINE DBI.G.SECURITY.RECORD(YourFile,YourRecord,YourId,ErrorFlag)**

YourFile	String representing the filename to check
YourRecord	The record to check against the entity security system
YourId	The Id of the Record to check
ErrorFlag	Returned by DesignBais. 1 = No Access 2 = Read Only Access

Eg: CALL DBI.G.SECURITY.RECORD("DBCLIENT",MYRECORD,MY.ID,ERROR.FLAG)

This will check all Entity Security definitions that apply to "YourFile". If any field within "YourRecord" that is defined in any definition is restricted by the Entity Security definition then ERROR.FLAG will be set.

- ERROR.FLAG = 0 – all fields are unrestricted for the user and the user group
- ERROR.FLAG = 1 – at least one field has access denied for the user or the user group
- ERROR.FLAG = 2 – at least one field has read only access for the user or the user group

When calling DBI.G.SECURITY.RECORD the developer can pass in a list of User Groups that apply to the user via the "ErrorFlag" argument. To do this use the following syntax:

ERROR.FLAG<1> = 'GROUP.LIST'
ERROR.FLAG<2> = multivalued list of User Groups

ERROR.FLAG is interrogated within the subroutine and the list of user groups is extracted. ERROR.FLAG is then reset to 0.

DBI.G.SECURITY.LIST

Is used to check a supplied list of record identifiers against the Entity Security System and return a list that is valid for that user.

You may have a situation where a particular user can only view particular entity members. An example might be that a particular user can only see certain geographical territories.

If this instance the security list routine would be called to reduce the list of geographical territories.

Typically, if the application is using DesignBais Entity Security, this routine would not be used. There are instances though where DesignBais may not be providing the list of entity members directly.

Usage: **SUBROUTINE DBI.G.SECURITY.LIST(YourFile,YourListIn,ReturningList,YourListDelimiter)**

YourFile	String representing the filename to check
YourListin	The list of record identifiers to be checked against Entity Security
ReturningList	The list of record identifiers that the Entity Security System has returned
YourListDelimiter	The Delimiter used in the YourListIn and ReturningList variable

Eg: CALL DBI.G.SECURITY.LIST(“DBCLIENT”,MYLISTIN,RETURNEDLIST,AM)

DBI.G.SORT.ONFORMREPORT

This routine is used to sort an On-form Report on a DesignBais form.

Typically this would be called in response to a click event occurring on Row zero of a report.

Usage: **CALL DBI.G.SORT.ONFORMREPORT(REPORTNAME,COLUMNTOSORT,SORTJUSTIFICATION)**

ReportName	The report name as it appears on the form
ColumnToSort	The number representing the column on the report to sort.
SortJustification	The justification required for the sort.

R	= Right Justified {Numeric}
L	= Left Justified {Alpha}
DR	= Date Right Justified
DL	= Date Left Justified
TR	= Time Right Justified
TL	= Time Left Justified

Subsequent sorts on the same column will reverse the direction.

The common variable **DBLASTDIRECTION** contains Last Direction *vm* Last Column *vm* Report Name.

Last Direction is either *A* or *D*.

You may modify this to stop the standard behaviour of reversing subsequent sorts.

Example:

```
*
REPORT:
*
    ROW = FIELD(DBREPORT.CELL,".",1)
    COL = FIELD(DBREPORT.CELL,".",2)
    BEGIN CASE
        CASE EVENTSOURCE = "R.REPORT1"
            IF ROW=0 THEN
                SORTJUST='L'
                BEGIN CASE
                    CASE SCREEN.NO = "D10"
                        IF COL = 1 OR 10 THEN SORTJUST = 'R'
                    CASE SCREEN.NO = "D20"
                        NULL
                END CASE
                CALL DBI.G.SORT.ONFORMREPORT(EVENTSOURCE,COL,SORTJUST)
                RETURN
            END
    END CASE
    RETURN
```

DBI.G.RESEQ.FORMDATA

This routine is used to sort fields and reset XML labels on a DesignBais form.

Typically this is necessary when creating a form at run time and writing it to the DBISESSIONS file and employing DBFORMLOCAL to run the form.

Usage: **CALL DBI.G.RESEQ.FORMDATA (SortParameter,YourFormRecord)**

SortParameter	<1> Attribute 1 must be null.
	<2> Sort By Field Name. See below.
YourFormRecord	The modified form record that is to be written to DBISESSIONS.

After a developer modifies a form at run time it should be passed into this subroutine before it is written to DBISESSIONS and the form id is added to DBFORMLOCAL. (see DBFORMLOCAL).

This routine is required to ensure that the XML labels on the form are set correctly after it is modified by the developer. Call this routine after modifying the form and before it is written to DBISESSIONS.

Sort By Field Name

This parameter must be null to set the standard DesignBais form field sequence. This sorts fields by:

TABINDEX 'R%4':ROW 'R%5':COL 'R%5':ROWSPAN 'R%5':COLSPAN 'R%5'

The following options are available, if required for use with DBFORMLOCAL for example, but should not be used if the developer is saving the form to the DBIFORMS file where the DesignBais standard sort sequence is recommended:

- DBIF.FIELD.PROPERTY Field Property (TEXT, BUTTON, INPUT, REPORT, GRAPH, CHECK, RADIO)
- DBIF.FIELD.NAME.LIST Field Name (the field name used in DBIPROP Ids)
- DBIF.FIELD.ATTR Field Attribute value
- DBIF.FIELD.AFTER Field Process After (either a subroutine name or a form name)
- DBIF.FIELD.READ.USE.LIST Read variable name (DBRECORD, DBOTHER.RECORD(n))
- DBIF.FIELD.TEXT.LIST Field Text
- DBIF.FIELD.SECTION Section Name

DBI.G.EXCLUSIVE

This routine is used to check to see if a record has a DesignBais Exclusive lock set.

Usage: **SUBROUTINE DBI.G.EXCLUSIVE(YourRecordPath,YourRecordToCheck,Locked,LockedUserRec)**

YourRecordPath This varies depending on the database flavour. You must specify the correct RecordPath format.

UniVerse

```
STATUS FILEVAR FROM F.FILENAME ELSE
        *Error Message
END
*
FILEVAR = FILEVAR<27>
```

UniData

```
FILEVAR = FILEINFO(F.FILENAME,2)
```

Other DBMS Platforms

```
FILEVAR = DBIACCOUNT:">":YourFilename
```

YourRecordToCheck

The record ID (key) to check

Locked Set to one (1) if the record is locked via a DesignBais Exclusive lock.

Set to zero (0) if the record is not locked via a DesignBais Exclusive lock.

LockedUserRec Returns the ID of the user locking the record, and the date and time the lock occurred.

<1> = User ID
<2> = Date (Internal Format)
<3> = Time (Internal Format)

Note regarding the way exclusive locks are held by DesignBais:

Locks are kept on DBISESSIONS which is opened as F.DBISESSIONS.MAIN in COMMON.

The DBISESSIONS Id is *RecordPath*|*RecordID*.

The locks are also kept per user on DBISESSIONS Id of *L*|*WEBLOGON*.
There are 4 multi-valued attributes:
<1> = MV SESSION.IDs
<2> = MV associated with <1> = The lock *RecordPath*|*RecordID* in a sub-valued list.
<3> = MV associated with <1> = DATE locked in a sub-valued list.
<4> = MV associated with <1> = TIME locked in a sub-valued list.

In the event of a problem where an exclusive lock cannot be released arrange for all other users to exit any exclusive locks and then clear the DBISESSIONS records referenced above. All references should be cleared. Exclusive locks for a user are displayed in the Exclusive Locks form DBIUSERS_D60 by clicking the *Show Locks For My User Only* link.

It is also necessary to check the LIST.READU in case someone is editing the record.

DBI.G.MG

This Subroutine can be called from any subroutine and will provide a list of forms/reports that a user has access to.

Usage:

CALL DBI.G.MG (*User Id, Returned Menu Items, Type*)

User Id - DesignBais User Id.
Returned Menu Items<1> - A list of menu processes (Including any modifiers ~M ~L ~E ~S)
<2> - Associated descriptions

Type should always be GET

Eg.

CALL DBI.G.MG (myuser,MYMENUS,"GET")

Print Thru (only in Version 6 and below)

The Print Thru function allowed existing reports to print directly through to the client printer.

This function is no longer available in DesignBais Release 7 and above.

This prints the report without modifying any of the file content.

The following text details step-by-step instructions on how to invoke the Print Thru process.

Step 1.

You must ensure that the file to be printed can be seen by the webserver. This means that the file has to be on the webserver or must be in a virtual directory that the webserver has access to

Step 2.

Build the URL to invoke the print thru routine.

WEB1 = CHANGE(WEBSERVER,”webRemote.Asp”,”dbprintX.asp”) ; Eg http://dbqa/db/dbprintX.asp

DIRECTORYPATH = *Path to Document*

PRINTERPATH = *Path to Printer*

URL = WEB1:”?j=”:SESSION.ID:”%26u=”:DIRECTORYPATH:”%26p=”:PRINTERPATH :”;newWindow;Printing;height=180px,width=420px"

Eg.
http://dbqa/db/dbprintX.asp?j=767656%26u=/printout/xyz.txt%26p=\\BAIS01\JimPrinter;newWindow;Printing;height=180px,width=420px"

Step 3.

DBCALLURL = URL Instruct DesignBais to invoke the URL

A progress window will appear notifying the user of printing progress.

The printer path must must include the printer name exactly as it is named in Windows. The name is case sensitive. The printer path would typically be //machine_name/printer_name.

DBI.P.CALLDBSUB

This is a program to initialise common and call a DesignBais subroutine. Some database platforms prevent the calling of a subroutine in the phantom command. This program can be called in order to call a subroutine. The logic can then intercept this within a new event “PHANTOM”.

It is useful when there is a requirement to process a phantom job and the routine to be processed by the phantom is part of a subroutine as opposed to a program. One use for the DBI.P.CALLDBSUB program is to facilitate the calling of a DesignBais subroutine normally used for basic process calls for DesignBais forms. Adding the ‘PHANTOM’ event to the subroutine’s existing Event Table CASE statements provides a hook to call a section of code terminated with the STOP command if necessary. By this means the code used by the phantom processing can be in the same routine that is used for all subroutine process calls associated with a DesignBais form.

DBI.P.CALLDBSUB is run with 3 arguments on the command line:

DBI.P.CALLDBSUB Sub.Name EventSource EventParameter

Sub.Name	= DesignBais subroutine to call
EventSource	= Becomes PROCESS.EVENTSOURCE
EventParameter	= Becomes PROCESS.PARAMETER

This program will set the PROCESS.EVENT to “PHANTOM”.

The following example demonstrates its use:

RUN DBI.P.CALLDBSUB SUBR.NAME SUB.EVENTSOURCE SUB.PARAMETER

This will result in the common variables set as follows followed by the call to the subroutine SUBR.NAME:

```
PROCESS.EVENT = "PHANTOM"
PROCESS.EVENTSOURCE = SUB.EVENTSOURCE
PROCESS.PARAMETER = SUB.PARAMETER
CALL @SUB.NAME
STOP
```

DBI.P.UPDATE.FORMS.CONTROL.KEY

This program (run from TCL), will process each form and set-up the Control+Enter button event for each button that has a Return to Field nominated. The actual return to field will then be flagged with the Control+Enter event.

This will not set-up the Control+Enter event for any button that has been nominated as a Submit, Delete or Clear button regardless of whether they have a return to field nominated.

DBI.G.OVERLAY

This subroutine displays an overlay form. The base form must have the “Overlay Required” check box ticked in order for the call to DB.G.OVERLAY to be processed. The following code snip provides an example of how this subroutine can be invoked.

The position of the overlay form is controlled by the POSITION.FLAGS argument which can have up to 5 values as follows:
<1,1> - Position From
<1,2> - Position Type
<1,3> - Relative Shift
<1,4> - Absolute col position (with row position will override first 3 options if present)
<1,5> - Absolute row position (with col position will override first 3 options if present)

Position Type of P indicates the the col and row will be taken from the field on the form that has the After Field process parameter equal to the value in Position From.

Any other value in position type indicates that Position From is either a field name, a check box group name, or an field xml label reference. The position will be derived from the col and row position of the named element on the form.

The relative shift is controlled by the first character of Relative Shift. The remaining characters of Relative Shift contain the col and row positions RCOL and RROW separated by a comma. Assume the position of the form element that is controlling the relative position is FCOL and FROW with associated spans of FCOLSPAN and FROWSPAN then:

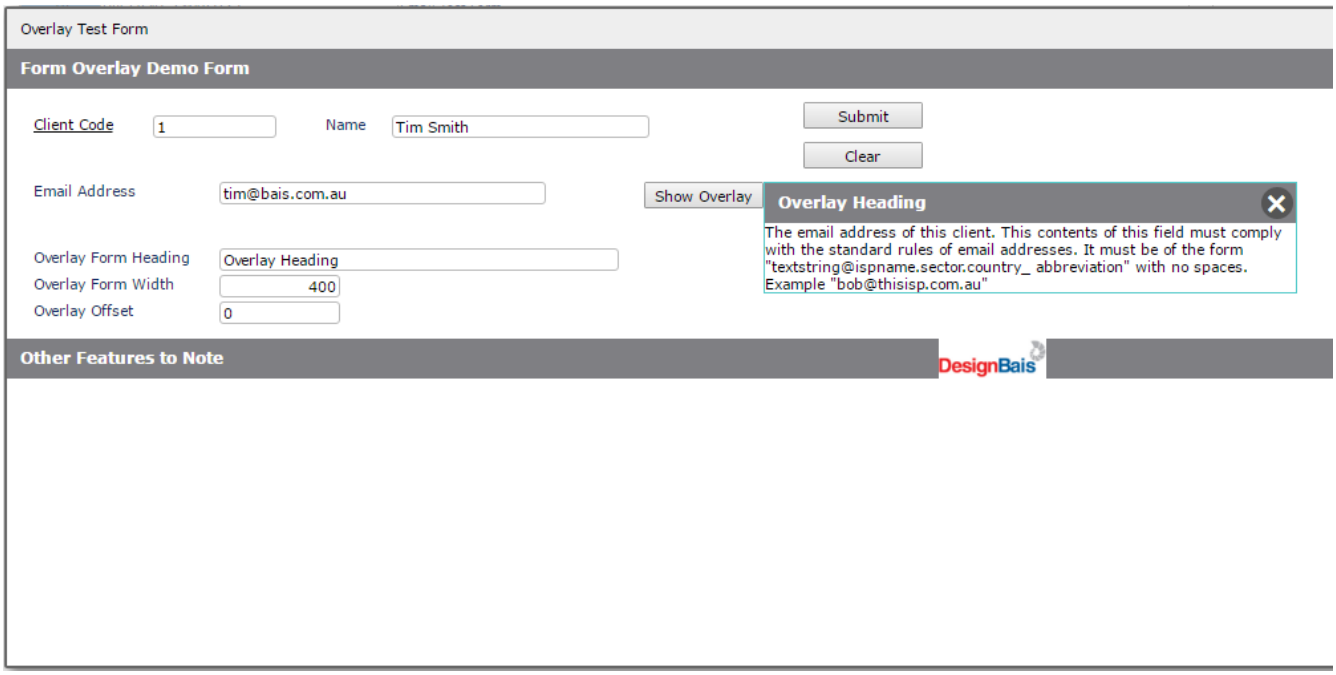
Relative Shift Parameter	Relative Shift
L	FCOL – WIDTH.FLAG – RCOL - FCOLSPAN
R	FCOL + RCOL + FCOLSPAN
A	FROW – WIDTH.FLAG + RROW – (FROWSPAN*2)
B	FROW + WIDTH.FLAG + RROW + FROWSPAN

WIDTH.FLAG can be negative.

```
*
READ.HELP.TEMPLATE:
*
! in this example the overlay form will be positioned relative to the position of THIS.FIELD
      THIS.FIELD = 'DBC.EMAIL'
      LINE.PARAM = 'R5,5’
      READV TEXT FROM F.DBIPROP,'DBCLIENT*’:THIS.FIELD',DBIP.HELPTEXT ELSE TEXT = 'Help text is missing'
! The value in HTML.TAGS value 1 is used as the heading text in the header bar of the overlay form
      HTML.TAGS  = DBRECORD<DBC.OVERLAY.HD>
      WIDTH.FLAG = 300
      GOSUB DISPLAY.HELP
      RETURN

*
DISPLAY.HELP:
*
      FORM.ID = FIELD(SCREENROOT,'_',1):'*':SCREEN.NO

! read the form record to get the style group then read the style group to get the label header class

      READ FORM.REC FROM F.DBIFORMS,FORM.ID ELSE FORM.REC = ""
      FORM.STYLE.GROUP = FORM.REC<DBIF.STYLE.GROUP>
      READ STYLE.GROUP.REC FROM F.DBISTYLEGROUP,FORM.STYLE.GROUP ELSE STYLE.GROUP.REC = ""
      USE.CLASS = STYLE.GROUP.REC<DBISG.DEFAULT.LABEL.HEADER>
      IF USE.CLASS = "" THEN USE.CLASS = "dbTitleBar"

! massage the help text for the field

      TEXT = CHANGE(TEXT,"'",'&#39;')
      TEXT = CHANGE(TEXT,'"','&#34;')
      TEXT = CHANGE(TEXT,"-",'&#45;')
   IF TEXT = '' THEN TEXT = 'Help not available for this activity'
!
      HTML.TAGS<1,2>  = "" ; * Reserved for image
      HTML.TAGS<1,3>  = TEXT
```

```
!
      FRAME.TAGS      = "style=overflow:hidden;border-style:solid;border-width:1px;border-
color:#55CACA;width:":WIDTH.FLAG:"px"
!
      STYLE.TAGS      = 'style="background-color:#55CACA;width:99%"'   ;*
      STYLE.TAGS      = 'class= "':USE.CLASS:'"'
      STYLE.TAGS<1,2> = 'style="text-align:right;"'
      STYLE.TAGS<1,3> = 'style="width:100%"'
!
      POSITION.FLAGS  = THIS.FIELD:VM:"F":VM:LINE.PARAM:VM:VM
!
      CLOSE.IMAGE     = ''
      CALL DBI.G.OVERLAY(FRAME.TAGS, HTML.TAGS, STYLE.TAGS, CLOSE.IMAGE, POSITION.FLAGS, WIDTH.FLAG)
      RETURN
```

Example Overlay Form: DBDEMO*OVERLAYTEST



This form can be run in the DBINET.DEMO account and the code can be viewed in the routine DBI.I.DEMO in DBLIB.

DBI.G.FOLD.TXT

This subroutine ‘folds’ text to a specified line length.

CALL DBI.G.FOLD.TXT(TEXT,FOLD.LEN)

TEXT String to be folded to length. The string can be multi-attributed or multivalued and can contain sub values.

FOLD.LEN The required length of each line.

The resultant folded text is returned as a multivalued string in TEXT.

The content of TEXT that is passed in is treated as one long string. It is broken into lines of maximum length FOLD.LEN but the line break is at the space character located before but closest to the specified length.

DBI.G.DYNAMIC.FORMNET

This subroutine can be used to insert form elements into an existing form record on DBIFORMS.

CALL DBI.G.DYNAMIC.FORMNET(THISACTION,FIELDLIST)

THISACTION String to be folded to length. The string can be multi-attributed or multivalued and can contain sub values.

The valid values are:

INSERT	The fields in FIELDLISTwill be inserted before all other existing fields.
INSERTPOS:position	The field elements in FIELDLIST will be inserted at the *position* multivalue. If *position* is a field name on the form being updated then the field elements in FIELDLIST will be inserted before the multivalue position of this field on the target form.
APPEND	The fields in FIELDLISTwill be appended after all other existing fields.
DELETE	The fields named in FIELDLIST attribute DBIF.FIELD.NAME.LIST will be deleted. The other attributes of FIELDLIST are not used.
DELETEPOS	The fields in the positions defined in FIELDLIST<1> will be deleted. These positions must be specified in ascending sequence.

FIELDLIST The details of the form elements to be added to the form in the same attribute positions as a form record on DBIFORMS.

For DELETE only the field names in DBIF.FIELD.NAME.LIST are required.

For DELETEPOS attribute 1 of this argument must be a list of field multivalue positions in ascending sequence passed as a multivalued string.

The following example demonstrates how to programmatically add the Form Help button to a form in UniVerse using equate names for the DBIFORMS attributes. It is important to save and restore TABLEDATA if you are processing within a DesignBais session so as to avoid corrupting the form from which you are calling your basic code.

```
FIELDLIST = ''
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.COL.LIST,1,0,FORM.WIDTH-30)
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.ROW.LIST,1,0,5)
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.COL.SPAN.LIST,1,0,20)
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.ROW.SPAN.LIST,1,0,20)
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.LENGTH.LIST,1,0,'')
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.DISABLED,1,0,'N')
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.TEXT.LIST,1,0,TEXT.LABEL)
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.NAME.LIST,1,0,HELP.BUTTON.NAME)
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.TYPE.LIST,1,0,"ALPHA")
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.SECTION,1,0,"Help")
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.XML.LABEL,1,0,INS.LABEL)
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.PROPERTY,1,0,"BUTTON")
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.CLOSE.MODAL,1,0,"N")
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.TAB.INDEX,1,0,"0")
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.DISPLAY.CLASS.LIST,1,0,DISPLAY.CLASS)
FIELDLIST = INSERT(FIELDLIST,DBIF.FIELD.AFTER,1,0,"DBI.G.BUTTON")
*
MATBUILD SAVE.TDATA FROM TABLEDATA
MATPARSE TABLEDATA FROM FORMREC,AM
*
THISACTION = "INSERTPOS:":INS.POS
CALL DBI.G.DYNAMIC.FORMNET(THISACTION,FIELDLIST)
*
MATBUILD FORMREC FROM TABLEDATA
*
```

MATPARSE TABLEDATA FROM SAVE.TDATA,AM

DBI.G.CLER

This subroutine:

- loads dictionaries for a file or files into the DBIPROP file; performs a dummy amend field to regenerate the DBIFILES record
- builds equate items for the nominated file(s)
- rebuilds dictionary items for a file or files, from the DBIPROP field properties
- creates DBIFILES record for file(s) if not already present

CALL DBI.G.CLER(FILE.LIST,OPTIONS)

FILE.LIST

FILE.LIST<1>	multivalued list of file names
FILE.LIST<2>	multivalued list of file descriptions
FILE.LIST<3>	multivalued list of equate file names
FILE.LIST<4>	multivalued list of equate prefix names

OPTIONS

A string containing any combination of L, E or R

L	load dictionaries into DBIPROP
E	build equates item
R	rebuild dictionaries from DBIPROP

DBIFILES records will only be created or if no DBIFILES item is pre-existing.

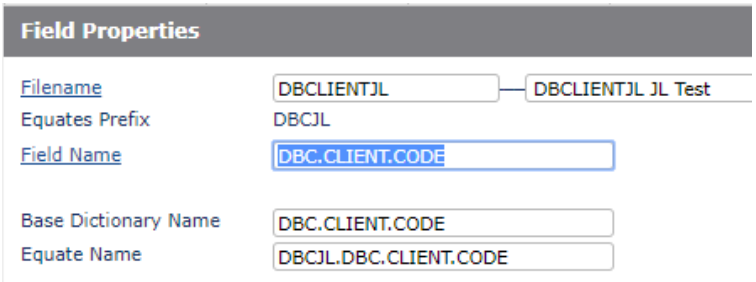
This subroutine invokes the standard calls within DBI.I.DBIPROP to perform the updates. Common variables used during calls to DBI.I.DBIPROP will be saved before running DBI.G.CLER and re-instated afterwards.

Example:

FILE.LIST = “”
FILE.LIST<1> = “MYFILE1”:VM:”DBCLIENTJL”
FILE.LIST<2> = “Test File 1”:VM:”JL Test”
FILE.LIST<3> = “DBEQU”:VM:”DBEQU”
FILE.LIST<4> = “MF1”:VM:”DBCJL”

CALL DBI.G.CLER(FILE.LIST,”LE”)

This will create DBIFILES records for the files MYFILE1 and MYFILE2 if they do not exist.
The dictionary items of both files will be loaded into DesignBais DBIPROP file.
The Base Dictionary Name will be the same as the source dictionary id. The Equate Name will be formed by concatenating the specified equate prefix name “DBCJL” to the name of each source dictionary record as shown in the example below for the DBCLIENTJL file.



DBI.G.AJXCMD

This subroutine generates ajax javascript commands and adds them to the common variable DBAJAXCMD from where they will be sent to the browser.

CALL DBI.G.AJXCMD(AJX.FUNC,AJX.ARG)

Functions prefixed with “rd” are designed to be used with the responsive design release of DesignBais.
Refer to the DesignBais Responsive Design Manual for additional details on all DBAJAXCMD commands.

AJX.FUNC

The following functions are available. Abbreviation is permitted as shown [n/a indicates no abbreviation]:

Function	*Abbreviation*	*Description*	*Notes*
rdHide	HD	Hide an element on an RD form	RD forms only
rdShow	SH	Show an element on an RD form	RD forms only
rdSetText	ST	Set text in an element on an RD form	RD forms only
rdSetVal	SV	Set the value in an element on an RD form	RD forms only
rdSetStyle	SS	Set the style of an element on an RD form	RD forms only
rdSetAttribute	SA	Set an attribute	RD forms only
rdRemoveAttribute	RM	Remove an attribute	RD forms only
rdlookupJSON	LJ	JSON lookup	RD forms only
rdShowConfirm	SC	display a "confirm” dialog	RD forms only
rdShowModal	SM	displays an RD row as a modal form	RD forms only
rdHideModal	HM	hides an RD row displayed as a modal form	RD forms only
rdAlertBox	AB	display message using browser's native alert box	RD forms only
rdShowMessage	SMSG	display a standard message panel on the page	RD forms only
rdHideMessage	HMSG	hide a standard message panel on the page	RD forms only
rdHideMessageAll	HMSGA	hide all message panels on the page	RD forms only
rdScrollToID	STID	scroll page to bring the element row into view	RD forms only
rdShowAllRows	SAR	display all rows of header, footer or body if hidden	RD forms only
rdHideAllRows	HAR	hide all rows of header, footer or body if displayed	RD forms only
rdShowSection	SSN	show header, footer or body	RD forms only
rdHideSection	HSN	hide header, footer or body	RD forms only
rdSetElementValue	SEV	directly set the value of any element	RD forms only
rdNavigate	NAV	navigate to the specified page	RD forms only
rdDeleteTableRow	DTR	delete a row from a table (refer Responsive Design Manual for syntax)	RD forms only
jqsv	JQSV	jquery set value	RD and non-RD forms
jqst	JQST	jquery set text	RD and non-RD forms
jqshtml	JQSHTML	jquery set html	RD and non-RD forms
jqsh	JQSH	jquery show()	RD and non-RD forms
jqhd	JQHD	jquery hide()	RD and non-RD forms
jqsa	JQSA	jquery set attribute	RD and non-RD forms
jqra	JQRA	jquery remove attribute	RD and non-RD forms
jqss	JQSS	jquery set css	RD and non-RD forms
addCode	AC	add code	RD and non-RD forms
addScript	AS	add script	RD and non-RD forms
addCSS	CSS	add CSS to the DesignBais style sheet	RD and non-RD forms
setSlider	SL	create a slider control	RD and non-RD forms
dbslidepanel	SP	create a slide panel	RD and non-RD forms
dbCarousel	CAR	create an image carousel	RD and non-RD forms
addEvent	AE	add event	RD and non-RD forms
customAttribute	CA	set custom attribute at run time	RD and non-RD forms
removeCustomAttribute	RCA	remove custom attribute at run time	RD and non-RD forms
dbDayPicker	DP	day (date) picker	RD forms only
dbDayPickerSetHTML	DPSH	day (date) picker	RD forms only
showSpinner	SSPN	show spinner text	RD and non-RD forms
hideSpinner	HSPN	hide spinner text	RD and non-RD forms
authenticated	AUTH	authenticate user access to *uploads* folder	RD and non-RD forms
setSelectionRange	SSR	set the start and end selection for input field	RD and non-RD forms
ajaxAsync	n/a	set asynchronous/synchronous mode	RD and non-RD forms

AJX.ARG
This is multiattributed, multivalued, and multisubvalued in <2>, <3> and <4>.

Example rdSetStyle:

```
IF DBVALUE = 'N' THEN
   CLR1 = 'thistle'; CLR2 = 'yellow'; WID3 = 'thick'
END ELSE
   CLR1 = '#8e578e'; CLR2 = '#ff4d4d'; WID3 = 'thin'
END
AJXCMD = 'SS';* rdSetStyle
AJX.ARG = ''
AJX.ARG<1,-1> = 'd948937'
AJX.ARG<2,-1> = 'element'
AJX.ARG<3,-1> = 'background-color':SVM:'border-width'
AJX.ARG<4,-1> = CLR1:SVM:WID3

AJX.ARG<1,-1> = 'd978206'
AJX.ARG<2,-1> = 'element'
AJX.ARG<3,-1> = 'color':SVM:'background-color':SVM:'border-width'
AJX.ARG<4,-1> = 'red':SVM:CLR2:SVM:WID3
CALL DBI.G.AJXCMD(AJXCMD,AJX.ARG)
```

Note that when using the function *jqsv* in non-RD forms the developer should use the field property name rather than the xml label id since the xml label id will vary based on:

- whether the form has a header form or not
- if the form is modified then existing xml label ids may change

Example:

```
AJXCMD = “JQSV”
AJX.ARG = “”
AJX.ARG<1,1> = ”DBC.WORK2.WK”
AJX.ARG<2,1> = “element”
AJX.ARG<3,1> = “”
AJX.ARG<4,1> = “new field value”
CALL DBI.G.AJXCMD(AJXCMD,AJX.ARG)
```

Example: updating a slide panel where the slide panel content is in a field called DEM.WORK4.WK:

```
AJX.ARG = 'DEM.WORK4.WK'
JMAX = DCOUNT(DBWORK<DEM.TEMP.FIELD.WK>,VM)
FOR J=1 TO JMAX
   IF DBWORK<DEM.TEMP.FIELD.WK,J> # '' AND DBWORK<DEM.TEMP.VALUE.WK,J> # '' THEN
      AJX.ARG<3,1,-1> = DBWORK<DEM.TEMP.FIELD.WK,J>
      AJX.ARG<4,1,-1> = DBWORK<DEM.TEMP.VALUE.WK,J>
   END
NEXT J
CALL DBI.G.AJXCMD('SP',AJX.ARG)
 * Show the slidePanel
AJX.ARG = ''
AJX.ARG<1,-1>  = 'slidePanel'
AJX.ARG<2,-1>  = 'element'
CALL DBI.G.AJXCMD('SH',AJX.ARG)
```

Refer to the DesignBais Responsive Design manual for a description of how to implement a slide panel on both RD and Non-RD forms.

Example: Add Event

This example shows how to add the “onmousedown” and “onblur” events to a field with a dropdown selection list.

If the script in AJX.ARG<3> does not start with “function(“, as in this example below, then DesignBais prepends “function(e){“ and appends “}”.

The “onmousedown” event sets the size of the dropdown to be equal to the number of elements in the list.

```
 * Set Dropdown Size when clicked
    AJX.FUNC = "addEvent"
    AJX.ARG    = "DBIPM.U.ID.WK"
    AJX.ARG<2> = "onmousedown"
    AJX.ARG<3> = 'this.size=this.length;'
    *
    AJX.ARG<1,-1> = "DBIPM.U.ID.WK"
    AJX.ARG<2,-1> = "onblur"
    AJX.ARG<3,-1> = 'this.size=0;'
    CALL DBI.G.AJXCMD(AJX.FUNC,AJX.ARG)
```

The “onblur” event sets the size of the list to zero so that the list disappears after a selection is made as can be seen using the browser *Inspect* function:

Example: Open, Close and Toggle a slider panel using ajax command ‘SP’

The slider panel is displayed using the work field DEM.WORK4.WK. Refer to Demo Form DBDEMO_SLIDEPANEL.

```
CASE EVENTSOURCE = "B.OPEN"
   AJX.ARG = 'DEM.WORK4.WK'
   AJX.ARG<2> = 'open'
   CALL DBI.G.AJXCMD('SP',AJX.ARG)

CASE EVENTSOURCE = "B.CLOSE"
   AJX.ARG = 'DEM.WORK4.WK'
   AJX.ARG<2> = 'close'
   CALL DBI.G.AJXCMD('SP',AJX.ARG)

CASE EVENTSOURCE = "B.TOGGLE"
   AJX.ARG = 'DEM.WORK4.WK'
   AJX.ARG<2> = 'toggle'
   CALL DBI.G.AJXCMD('SP',AJX.ARG)
```

Example: Use the ‘CA’ command for RD forms where the custom attribute field is not available to the developer

```
AJX.ARG = element ID
AJX.ARG<3>      = attribute name
AJX.ARG<4>      = attribute value
CALL DBI.G.AJXCMD("CA",AJX.ARG)
```

Set the multi-select property of a select element:

```
AJX.ARG = 'BKT.PROP.Q47A'
AJX.ARG<3> = "multiple"
AJX.ARG<4> = "multiple"
CALL DBI.G.AJXCMD("CA",AJX.ARG)
```

Example: Move a form at runtime

Add this line to your basic subroutine to set the top edge of a form 170px down from the top of the window.

The *top* property sets or returns the top position of a positioned element. This property specifies the top position of the element including padding, scrollbar, border and margin. A positioned element is an element with the position property set to: relative, absolute, or fixed.

```
DBAJAXCMD<-1>='document.getElementById("dbBackGround':DBWLEVEL:'").style.top="170px";    '
```

Example: Day Picker

Example code is shown below.

```
*
VALIDATE:
*
      BEGIN CASE
         CASE EVENTSOURCE = "DBC.WORK1.WK" AND SCREEN.NO = "myscreen"
            DBDS<-1>='DBVALUE=':DBVALUE

            *** To select a date only see the code in AFTER.DISPLAY
            *** To apply your own HTML use the 4 lines below
            * AJX.FUNC = "DPSH"          ;* dbDayPickerSetHTML
            * AJX.ARG = "delivery"    ;* the ID of the <div> container in the RD page to be used for the day picker
            * AJX.ARG<3,1,-1> = '<div style="color:red;">hello</div>'
            * CALL DBI.G.AJXCMD(AJX.FUNC,AJX.ARG)
      END CASE
      RETURN
*
AFTER.DISPLAY:
*
      BEGIN CASE
         CASE SCREEN.NO = " myscreen"
            AJX.FUNC = "DP"            ;* dbDayPicker
            AJX.ARG = "delivery"    ;* the ID of the <div> container in the RD page to be used for the day picker
            AJX.ARG<3,1,-1> = 'startDate'               ; AJX.ARG<4,1,-1> = DATE()
            AJX.ARG<3,1,-1> = 'endDate'                 ; AJX.ARG<4,1,-1> = DATE()+90
            AJX.ARG<3,1,-1> = 'selectedDate'            ; AJX.ARG<4,1,-1> = '""'
            AJX.ARG<3,1,-1> = 'pickerTitle'             ; AJX.ARG<4,1,-1> = "Select delivery date and time"
            AJX.ARG<3,1,-1> = 'displayDate'             ; AJX.ARG<4,1,-1> = '27-Oct-2020'
            AJX.ARG<3,1,-1> = 'disabledArray'           ; AJX.ARG<4,1,-1> = '["14-Feb-2020","12-Feb-2020"]'
            AJX.ARG<3,1,-1> = 'customAttributeArray'    ; AJX.ARG<4,1,-1> = '[xxxx,9999]';* as required
            * myhiddenfield is the ID of the hidden field added to the RD page to pass data back to the database
            * AJX.ARG<3,1,-1> = 'onselect'              ; AJX.ARG<4,1,-1> = 'function ($el, d) {}'
            * AJX.ARG<3,1,-1> = 'onanimate'             ; AJX.ARG<4,1,-1> = 'function (b) { }'
            * AJX.ARG<3,1,-1> = 'hiddenID'              ; AJX.ARG<4,1,-1> = 'myhiddenfield'
            CALL DBI.G.AJXCMD(AJX.FUNC,AJX.ARG)
      END CASE
      RETURN
```

The disabledArray contains a list of dates, as required, that display but cannot be selected
The customAttributeArray can be used as required.

Example:

Use the 'SSPN' (or 'showSpinner') command to display text above the server busy spinner

```
AJX.ARG = "Dummy"           (element ID is not related to any element but cannot be null)
AJX.ARG<2>      = text to display
AJX.ARG<3>      = "true" or "false" (used to optionally blur and block the background)
CALL DBI.G.AJXCMD("SSPN",AJX.ARG)
```

Example:

Authenticate a user to access the website *uploads* folder.

```
AJX.FUNC        = “AUTH” or “authenticated”
AJX.ARG = “Y” or “y” or “TRUE or “true” or 1

CALL DBI.G.AJXCMD(AJX.FUNC,AJX.ARG)
```

Will set SESSION.REC<75> = “Y”. Set AJX.ARG to any other value will clear it.

Example:

Set selection range for a text area or input field.

```
AJX.FUNC        = “SSR” or “setSelectionRange”
AJX.ARG         = elementId1 :VM: elementId2
AJX.ARG<4,-1>   = start_pos1|end_pos1 :VM: start_pos2|end_pos2
CALL DBI.G.AJXCMD(AFX.FUNC,AJX.ARG)
```

Note:
elementId can be the field name on the form or the xml id. For Responsive Design it is the element Id.

The *start_pos* and the pipe separator can be omitted in which case the start range will default to zero.

Example:

Set asynchronous/synchronous mode for a form.

```
AJX.FUNC        = “ajaxAsync”
AJX.ARG         = “true” or “false”
CALL DBI.G.AJXCMD(AJX.FUNC,AJX.ARG)
```

The above call will place the command 'ajaxAsync=':AJX.ARG into the DBAJAXCMD common variable.

DesignBais runs with asyncronous ajax calls by default. **Synchronous mode has been deprecated by the browser companies**. However, developers may choose to run in synchronous mode until the browsers in use drop support for synchronous mode. This feature can be set in the GET SCREEN event slot by placing a subroutine name in the User or User Group *“Start Process (Form Load)”* with the appropriate call as shown:

Turn async mode on:

```
AJX.ARG = 'true'
CALL DBI.G.AJXCMD('ajaxAsync', AJX.ARG)
```

Turn async mode off:

```
AJX.ARG = 'false'
CALL DBI.G.AJXCMD('ajaxAsync', AJX.ARG)
```

To apply System or Global Parameter setting:

```
AJX.ARG = '' or AJX.ARG =’default’
CALL DBI.G.AJXCMD('ajaxAsync', AJX.ARG)
```

Developers can control this feature at any point if required, such as in a button event.

Example:

Using jquery set attribute *jqsa* with multivalue grid elements.

```
AJX.FUNC = 'jqsa'
AJX.ARG = 'DBC.INV.DATE'                        the multivalue field name
AJX.ARG<2> = 'placeholder'                      the attribute type to be set
AJX.ARG<4> = 'FRED ':J                          the attribute value
AJX.ARG<12> = 'I~|':J                           the grid part to be updated
CALL DBI.G.AJXCMD(AJX.FUNC,AJX.ARG)
```

In line with DBSTYLESPEC the grid parts in AJX.DATAIN<12> are:

```
B = grid border
R = grid data row
C = grid cell i.e. <td> component
I = Input or output element
```

The grid part can be followed by "~|row.col". For example:
"I~|3.7" target the Input/Output element in row 3 col 7

If no part is provided then the default is "I".

If no row id is specified then the change will be applied to the DBMVCOUNT row.

If no col is provided then col defaults to the position of the field name in the grid.

The above rules will be applied if the field provided in AJX.DATAIN<1> has an association name in TABLEDATA. This corresponds to form field property DBIF.FIELD.ASSOC.LIST.

The function can be applied to grid rows even if they are not initially displayed. But if the element specified is not available an undefined error will display. In this example placeholder is set for five rows:

```
FOR J=1 TO 5
   AJX.FUNC = 'jqsa'
   AJX.ARG = 'DBC.INV.DATE'
   AJX.ARG<2> = 'placeholder'
   AJX.ARG<4> = 'FRED ':J
   AJX.ARG<12> = 'I~|':J
   CALL DBI.G.AJXCMD(AJX.FUNC,AJX.ARG)
NEXT J
```

In the example below set the placeholder text for a field *PAY.REF* in the validation of field *PAY.BY*. The placeholder value is obtained from attribute 15 of the *PAY.MODE* file.

```
AJX.FUNC = 'jqsa'
AJX.ARG = 'PAY.REF'
AJX.ARG<2> = 'placeholder'
AJX.ARG<4> = OCONV(PAY.BY,"T$PAY.MODE;X;;15")
* AJX.ARG<12>     ;* not needed - DBMVCOUNT defines the row, field position defines the column
CALL DBI.G.AJXCMD(AJX.FUNC,AJX.ARG)
```

Place the call to DBI.G.AJXCMD in the MV_POST event slot. The placeholder text will then display when a new row is added to the grid. If you want to display the placeholder text in the first row of an empty grid when the form displays then the call can be added to the AFTER DISPLAY event. Once a row is displayed the validate event on a cell can be used to set the placeholder for subsequent cells in the row.

DBI.G.PDFREPORT

This subroutine converts a report to a PDF file.

Generate a report, save it to a cabinet drawer then use DBI.G.PDFREPORT to produce the PDF.

CALL DBI.G.PDFREPORT(CABINET.FILE,DRAWER.NAME,REPORT.NAME,PDF.PARAMS)

Arguments passed to this subroutine:

CABINET.FILE	The name of the cabinet file containing the report.
DRAWER.NAME	The name of the drawer within the specified cabinet.
REPORT.NAME	The name of the report in the specified drawer.
PDF.PARAMS	Pass in 3 attributes
	<1> From Page
	<2> To Page
	<3> Search String – locates this string in the report and the PDF starts at this point of the report.
	<4> optional path to the DBMail uploads folder – see example code below
	<5> the PDF file name which is passed to subroutine DBI.I.DBIPARMS via PROCESS.PARAMETER.

Arguments returned from this subroutine:

PDF.PARAMS	Returns the URL to call to access the PDF report.

Example:

Extract pages 11 to 21 of a report and convert to a PDF.

```
CABINET.FILE     = "DBCABINET1"
DRAWER.NAME      = "DRAWER|EmailTest"
REPORT.NAME      = "REP~18750-10099"
PDF.PARAMS       = 11
PDF.PARAMS<2>    = 21
CALL DBI.G.PDFREPORT(CABINET.FILE,DRAWER.NAME,REPORT.NAME, PDF.PARAMS)
DBCALLURL = PDF.PARAMS
```

Example code using PDF.PARAMS<4>.

DBI.G.PDFREPORT normally works through the web server i.e. via a normal DesignBais form. By setting a path in PDF.PARAMS<4> the report HTML is written to the DBMail folder. The call to DBI.G.DBMAIL then invokes the PDF conversion of an attachment.

```
* Test program
$INCLUDE DBI DBI.COMMON
$INCLUDE DBI E.DBIFORMS
$INCLUDE DBI E.DBIUSERS
$INCLUDE DBINET E.DBIPARMS
$INCLUDE DBINET E.DBIGLOBAL
$INCLUDE DBINET E.DBIMAIL
CALL DBI.G.INITVARIABLESNET
CALL DBI.G.OPENNET
CABINET.FILE = "BAGENCAB"
DRAWER.NAME = "DRAWER|PJUNE2020"
REPORT.NAME = "18.09:17:18"
PDF.PARAMS = ''
* the path to DBMail
PDF.PARAMS<4> = "E:\DBMAIL\ATTACHMENTS"
CALL DBI.G.PDFREPORT(CABINET.FILE,DRAWER.NAME,REPORT.NAME, PDF.PARAMS)
DBCALLURL = PDF.PARAMS
DBIACCOUNT = OCONV(@WHO,"MCU")                    ;* Used in DBMail xeml file ID
DBMAIL.REC = ''
DBMAIL.REC<DBM.MESSAGE>       = 'CONVERT TO PDF'   ;* Mandatory setting
DBMAIL.REC<DBM.FROM>          = 'DUMMY'            ;* Mandatory setting
DBMAIL.REC<DBM.ATTACHMENT>    = PDF.PARAMS
DBMAIL.REC<DBM.ATT.CONVERT>   = "PDF"
DBMAIL.REC<DBM.ATT.ZIP>       = "false"
DBMAIL.REC<DBM.PDF.LAYOUT>    = "P"
CALL DBI.G.DBMAIL(DBMAIL.REC)
CHECK = CHANGE(PDF.PARAMS,'.HTML','.pdf')
STM = 'DOS /C "dir E:\DBMAIL\ARCHIVE\':CHECK:'"'
CRT
PRINT STM
CRT
* We have to wait until DBMail completes the conversion
LOOP
   EXECUTE STM CAPTURING DETS RETURNING ERRS
   TST = INDEX(DETS,"File Not Found",1)
WHILE TST DO
   SLEEP 1
REPEAT
CRT 'DETS=':DETS
```

DBI.G.EMAILREPORT

This subroutine emails report pages to recipients.

The report must use @EMAILPRINT in order to assign pages to recipients.

Generate a report, save it to a cabinet drawer then use DBI.G.EMAILREPORT to email appropriate pages to each recipient.

CALL DBI.G.EMAILREPORT(CABINET.FILE,DRAWER.NAME,REPORT.NAME,EMAIL.PREF)

CABINET.FILE	The name of the cabinet file containing the report.
DRAWER.NAME	The name of the drawer within the specified cabinet.
REPORT.NAME	The name of the report in the specified drawer.
EMAIL.PREF	The integer corresponding to the email preferences dropdown selection list row or an alphanumeric value indicating the name of a DBMail template. Attributes 2 thru 11 can be used to set excel parameters and email attributes as described below.

Example:

CABINET.FILE = "DBCABINET1"
DRAWER.NAME = "DRAWER|EmailTest"
REPORT.NAME = "REP~18750-10099"
EMAIL.PREF = “”
CALL DBI.G.EMAILREPORT (CABINET.FILE,DRAWER.NAME,REPORT.NAME, EMAIL.PREF)

Note that DBMAIL must be configured and running if this routine is to work from a phantom.

PROCESS.PARAMETER may hold XLS options. Attribute 7 must contain the *Email To* address:

```
IF FIELD(DBCABINETREPORT,"-",3) = "XLS" OR THIS.PARAMETER<7>#'' THEN
      CALL DBI.G.CULTURE(3,RESULTS); * Get Culture, Table & Date Format from User, System or Global.
      EMAIL.XLS.DATE.FORMAT = RESULTS<1>
      EMAIL.XLS.CULTURE     = RESULTS<2>
      EMAIL.XLS.TABLE       = RESULTS<3>
      EMAIL.XLS.HAS.TABLE   = RESULTS<4>
      EMAIL.XLS.WKS         = "Sheet1" (requires a default value)
      * Override the settings from DBI.G.CULTURE is set via PROCESS.PARAMETER
      IF THIS.PARAMETER<2> # '' THEN EMAIL.XLS.DATE.FORMAT = THIS.PARAMETER<2>
      IF THIS.PARAMETER<3> # '' THEN EMAIL.XLS.CULTURE     = THIS.PARAMETER<3>
      IF THIS.PARAMETER<4> # '' THEN EMAIL.XLS.TABLE       = THIS.PARAMETER<4>
      IF THIS.PARAMETER<5> # '' THEN EMAIL.XLS.HAS.TABLE   = THIS.PARAMETER<5>
      IF THIS.PARAMETER<6> # '' THEN EMAIL.XLS.WKS         = THIS.PARAMETER<6>
      * Extract values passed via PROCESS.PARAMETER
      EMAIL.TO      = THIS.PARAMETER<7>
      EMAIL.SUBJECT = THIS.PARAMETER<8>
      EMAIL.MESSAGE = THIS.PARAMETER<9>
      EMAIL.FROM    = THIS.PARAMETER<10>
      EMAIL.TYPE    = THIS.PARAMETER<11>
      * Reset PROCESS.PARAMETER
      THIS.PARAMETER = THIS.PARAMETER<1>
```

DBI.G.CULTURE

This subroutine can be used to extract the culture, table format and date format for excel files from DesignBais User, System or Global parameters.

```
CALL DBI.G.CULTURE(RUN.TYPE,RESULTS)
```

RUN.TYPE	Set to 3 to extract xls file parameters.
RESULTS<1>	Date Format (either dd-MM-yyyy or MM-dd-yyyy)
RESULTS<2>	Excel Culture (defaults to “en-AU”)
RESULTS<3>	Excel Table Style
RESULTS<4>	Excel Format as Table
RESULTS<5>	Excel text encoding (ASCII or utf-8)

Notes:
Excel Culture - used by the conversion to Excel component to determine if an exported value is a date. The output format is determined by Date Format parameter. The user setting, if present, overrides the System Parameter which in turn, overrides the Global setting.

Excel Table Style - this setting determines if the data exported to Excel is to be displayed in table format and if so, the style of table. If not entered or Inherit is chosen then the setting will be Inherited from the Global setting. This setting may be overridden for individual users. If No Table Format is selected then this will mean that the raw data is exported to Excel with no table formatting.

Excel text encoding - This setting determines if the data exported to Excel is to be encoded as ASCII (default) or as utf-8. Using UTF8 together with the Culture setting allows Currency symbols to be applied when the Excel file is produced.

DBI.G.CABINETS

This subroutine can be used to purge eXpress cabinet drawers and to move reports between drawers.

DBI.G.CABINETPURGE

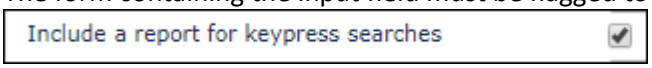
This subroutine can be used to purge eXpress cabinet drawers and to move reports between drawers.

```
    SUBROUTINE DBI.G.CABINETPURGE
* Purge eXpress Cabinet or move reports to another drawer
*
* Inputs are passed via common variable PROCESS.PARAMETER as a csv string
*
* In order to simplify the call to purge a cabinet only 1 input is required:
*    csv pos 1: CABINET.NAME     - the name of the Cabinet in which all drawers are to be purged
*
* Two additional inputs can be supplied if the default action is to be varied:
*    csv pos 2: CAB.ACTION       - PURGE or MOVE
*    csv pos 3: DELETEEMPTY      - 1 or Y if empty drawers (after PURGE) are to be deleted
*
* and all drawers in the cabinet will be purged based on normal cabinet report purge rules
*
* If the routine is called to move reports from 1 drawer to another then the name of the Cabinet containing the drawer that reports will be moved FROM
* is passed in csv pos 1
*
*    csv pos 4: DRAWER.FROM      - the name of the drawer that the reports will be moved FROM
*    csv pos 5: CABINET.NAME.TO  - the name of the Cabinet containing the drawer that reports will be moved TO
*    csv pos 6: DRAWER.TO        - the name of the drawer that the reports will be moved TO
```

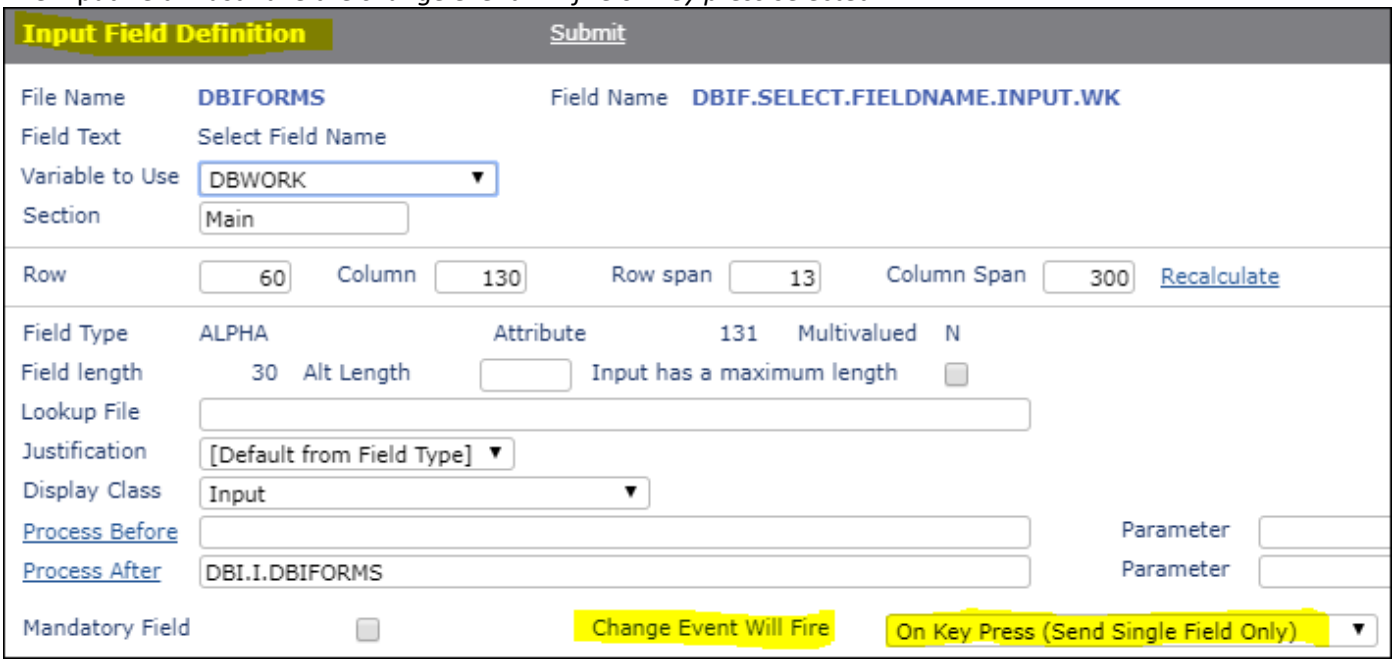
DBI.G.DISPLAY.DROP

This subroutine can be used to set up key-press activated dropdown list within an input field.

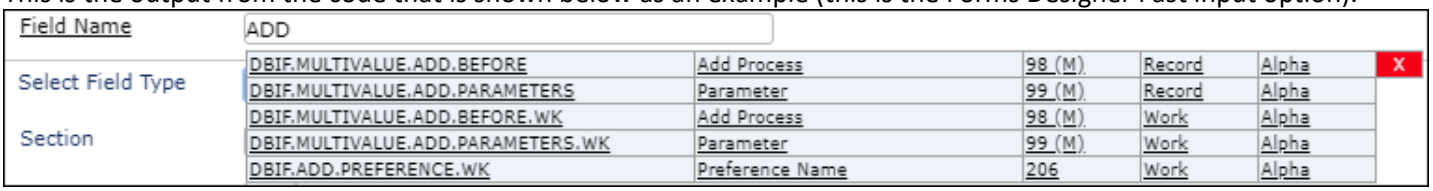
The form containing the input field must be flagged to include a report for keypress searches.



The input field must have the *change event will fire on key press* selected.



This is the output from the code that is shown below as an example (this is the Forms Designer Fast input option).



In this example the dropdown list is populated from the list of fields for the defined file held in the DBIFILES record in field DBIFI.REFRESH.LIST. See NAME.LIST below.

Use `CALL DBI.G.DISPLAY.DROP("","","","","",0,0,0,0)` to clear the report.

```
FAST.ITEM:
      DBKEYPRESSTOVALIDATE = 0
      IF DBVALUE = DBWORK<DBIF.SELECT.FILENAME.INPUT.WK> THEN
         RETURN
      END
      IF LEN(FAST.TEST) < 1 THEN
         CALL DBI.G.DISPLAY.DROP("","","","","",130,79,250,150)
         RETURN
      END
      READ DBIFILES.REC FROM F.DBIFILES,DBWORK<DBIF.SELECT.FILENAME.INPUT.WK> ELSE
         DBIFILES.REC = ""
      END
      NAME.LIST = DBIFILES.REC<DBIFI.REFRESH.LIST>
      RESULT.LIST = ""
```

```
NUMBER.OF.RESULTS = 0
FAST.TEST = OCONV(FAST.TEST,"MCU")
LOOP
   REMOVE TESTSTR FROM NAME.LIST SETTING NVTM ;*#EXCLUDE D3,OX
WHILE TESTSTR # "" OR NVTM DO
   TESTSTR = OCONV(TESTSTR,"MCU")
   IF INDEX(TESTSTR,FAST.TEST,1) THEN
      RESULT.LIST<1,-1> = TESTSTR
      NUMBER.OF.RESULTS += 1
   END
REPEAT
IF NUMBER.OF.RESULTS > 40 THEN
   NUMBER.OF.RESULTS = 40
END
HEADER.DETAILS = "Property":VM:"Screen Label":VM:"Field":VM:"Work":VM:"Type"
DISPLAY.DETAILS = ""
KEY.DETAILS = ""
JUST.DETAILS = ""
WIDTH.DETAILS = "38":VM:"28":VM:"10":VM:"10":VM:"10"
FOR DL = 1 TO NUMBER.OF.RESULTS
   THIS.RESULT = RESULT.LIST<1,DL>
   DBIPROP.ID = DBWORK<DBIF.SELECT.FILENAME.INPUT.WK>:"*":THIS.RESULT
   READ DBIPROP.REC FROM F.DBIPROP,DBIPROP.ID ELSE
      DBIPROP.REC = ""
   END
   OUTPUT.LINE = THIS.RESULT
   OUTPUT.LINE<1,2> = DBIPROP.REC<DBIP.SCREEN.LABEL>
   ATTRPOS = DBIPROP.REC<DBIP.FIELD.ATTRIBUTE>
   MVPOS = DBIPROP.REC<DBIP.FIELD.MULTIVALUE>
   SVPOS = DBIPROP.REC<DBIP.FIELD.SUBVALUE>
   FIELD.DESC = "*"
   BEGIN CASE
      CASE ATTRPOS AND MVPOS AND SVPOS
         FIELD.DESC = ATTRPOS:",":MVPOS:",":SVPOS
      CASE ATTRPOS AND MVPOS
         FIELD.DESC = ATTRPOS:",":MVPOS
      CASE ATTRPOS
         IF DBIPROP.REC<DBIP.MULTIVALUED> = "Y" THEN
            FIELD.DESC = ATTRPOS:" (M)"
         END ELSE
            FIELD.DESC = ATTRPOS
         END
   END CASE
   OUTPUT.LINE<1,3> = FIELD.DESC
   IF DBIPROP.REC<DBIP.WORK> = "Y" THEN
      OUTPUT.LINE<1,4> = "Work"
   END ELSE
      OUTPUT.LINE<1,4> = "Record"
   END
   BEGIN CASE
      CASE DBIPROP.REC<DBIP.FIELD.TYPE> = "N"
         TYPE.DESC = "Numeric"
      CASE DBIPROP.REC<DBIP.FIELD.TYPE> = "D"
         TYPE.DESC = "Date"
      CASE DBIPROP.REC<DBIP.FIELD.TYPE> = "T"
         TYPE.DESC = "Time"
      CASE 1
         TYPE.DESC = "Alpha"
   END CASE
   OUTPUT.LINE<1,5> = TYPE.DESC
   DISPLAY.DETAILS<DL> = OUTPUT.LINE
   KEY.DETAILS<DL> = STR(EVENTSOURCE:"|":THIS.RESULT:VM,6)
NEXT DL
CALL DBI.G.DISPLAY.DROP(HEADER.DETAILS,DISPLAY.DETAILS,KEY.DETAILS,WIDTH.DETAILS,JUST.DETAILS,130,79,650,135)
RETURN
```

DBI.G.SAVEFILE

Use this subroutine to save a record to a folder location.

Usage:

```
CALL DBI.G.SAVEFILE(SF.PATH,SF.CONTENT)
```

SF.PATH	The path to the location to which the file is to be saved.
SF.CONTENT	The record to be saved.

Example:

To save a record that has been created or manipulated within your subroutine to the web server *uploads* folder with a name prefixed with the database account name:

```
SF.PATH = “uploads/”:DBIACCOUNT:”_example.txt”
SF.CONTENT = YOUR.REC
CALL CALL DBI.G.SAVEFILE(SF.PATH,SF.CONTENT)
```

DBI.G.CM

Use this subroutine to set up a context sensitive menu for a selected form field.

There are 8 arguments:

CALL DBI.G.CM(MENU.ID,MENU.ITEMS,CONTAINER.STYLE,MENU.STYLE,XPOS,YPOS,DWIDTH,DHEIGHT)

MENU.ID - the name of a DesignBais menu located in the DBIMENUS file
MENU.ITEMS - LIST
CONTAINER.STYLE
MENU.STYLE
XPOS
YPOS
DWIDTH
DHEIGHT
Example:

The form field to which the context sensitive menu is linked requires a custom attribute. In the example below the field DBC.CLIENT.CODE has a custom attribute.
Use the *Custom Attributes* link to display the list of available custom attributes and select the *Context Menu* option:

oncontextmenu="getMouse(event);event.preventDefault();"

In the event handler section of your subroutine add the case:

```
    CASE THIS.EVENT = "CONTEXTMENU"
        GOSUB CONTEXTMENU
```

Add code for the paragraph CONTEXTMENU:

```
*
CONTEXTMENU:
*
    DBRETURN.TO.FIELD = "NOFOCUS"
    BEGIN CASE
        CASE EVENTSOURCE = "DBC.CLIENT.CODE"
            MENU.ID = "GARB1"
            MENU.ITEMS = ''
            CONTAINER.STYLE = 'dbaiscmContainer'
            MENU.STYLE = 'dbaiscmMenu'
            XPOS = 360
            YPOS = 55
            DWIDTH = 180
            DHEIGHT = 'auto'
            CALL DBI.G.CM(MENU.ID,MENU.ITEMS,CONTAINER.STYLE,MENU.STYLE,XPOS,YPOS,DWIDTH,DHEIGHT)

    END CASE
    RETURN
```

DBI.G.TSNAPNET

On return from a call to a layered form DBSTORE is not restored. Since DBSTORE is used to pass data around it retains its values.

Developers can however use a call to DBI.G.TSNAPNET(ID.TYPE) in the BUTTON event when invoking the a modal form using "~M". Set ID.TYPE to null.

Then DBRESTORE.BEFORE.MODAL = 1 restores DBRECORD, DBWORK and DBSTORE etc on return from the modal form.

Pre release 8.6.0.5 the snapshot keys on DBISESSIONS are of the form:

SCREENREC*SCREENROOT*SESSION.ID
SCREENREC-SS- SCREENROOT*ORIGINAL.SCREENROOT*SESSION.ID

From release 8.6.0.5 the DBISESSIONS records holding "SCREENREC" contents have a prefix of "SS" for all snapshots. This has been implemented in order to reduce the length of record ids. The SCREENREC*SCREENROOT*SESSION.ID will become:

SS* SCREENROOT*SESSION.ID

The old SCREENREC-SS-form_name prefix is now "SS-":DBWLEVEL*SESSION.ID. For example:

"SS-2*session.id" for a form displaying in level 2.

Chapter 15 – Glossary Maintenance

Glossary Maintenance

DesignBais utilizes a terms-based glossary to make the supporting of multiple languages much easier. As the glossary is terms-based, it is also very easy to modify specific terms for a user or client.

The glossary definitions are linked to the Glossary Names entered into the System Parameters form.

These glossary definitions are stored in four message types

1. **Screen Prompts**
   All text fields and labels used on a form

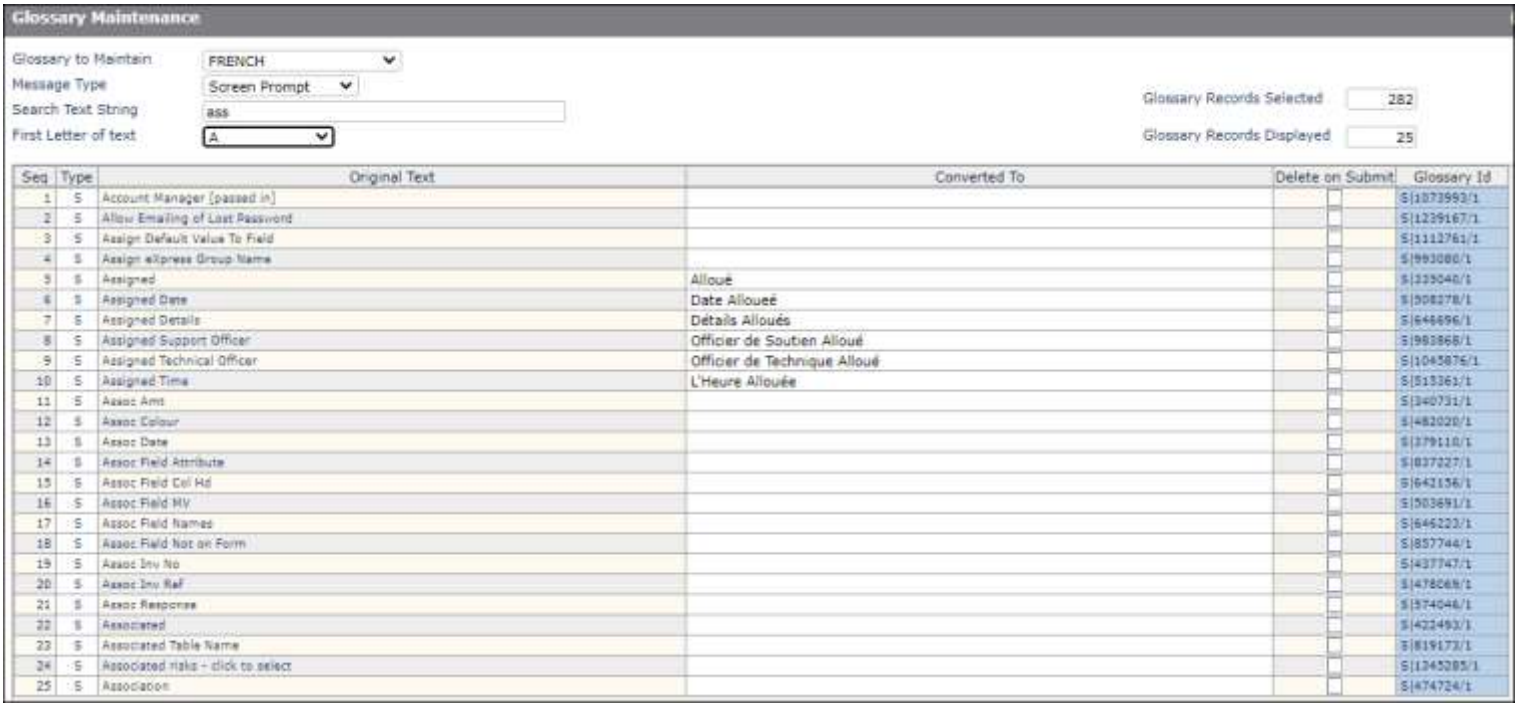
2. **Program Messages**
   Any program message text that is created via the DBI.G.GLOSSARY program. (See references to IERR.TEXT).

3. **Help Text**
   Any help text for Fields that resides in the Field Properties file DBIPROP.

4. **Menu Items**
   All menu text created using the Menu Definitions form.

All of these glossary types are stored in the file DBIGLOSSARY

Glossary Maintenance Form



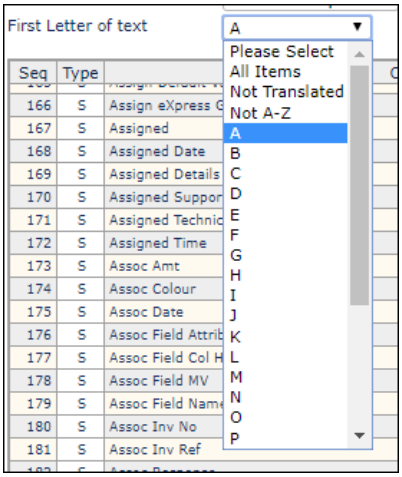
Prompts

Glossary to Maintain — Dropdown list containing all glossary names defined in the System Parameters. Select the glossary that you wish to maintain

Message Type — Dropdown list containing the message type to complete.

Search Text String — Enter a text string to use to filter the selection of glossary records. Base Glossary messages will be selected only if they contain this string. All three forms (upper case, lower case or text case) of this string will be applied.

First Letter of text — Contains a list of characters that are the start letter for each term or phrase. This mechanism helps to reduce the number of terms displayed at one time.



Original Text — Displays the original text. This cannot be modified.

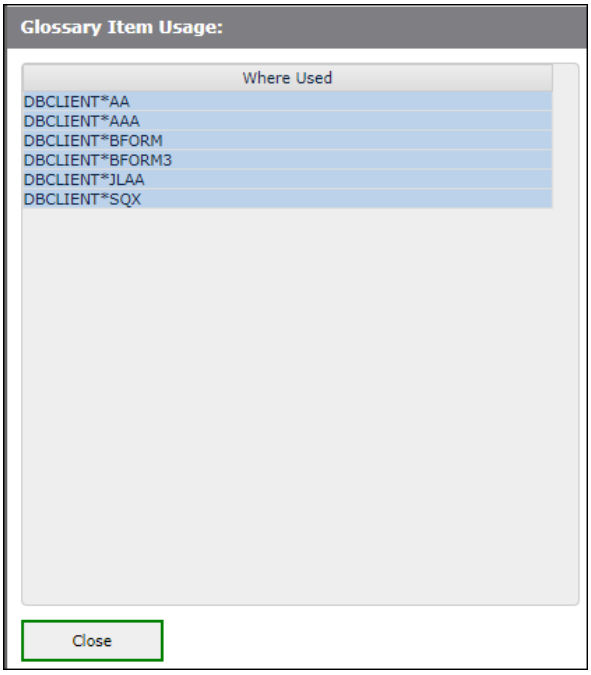
Converted to — The conversion for the required glossary.

Delete on Submit — There may be terms that are no longer used. Setting this prompt to 'Yes' will delete the entry from the glossary when the submit button is pressed.

Glossary Id — Click the Id in a selected row to display the glossary record in the *Maintain Glossary Record* form. The translated text for all glossaries, not just the glossary code entered in the Glossary to Maintain filter above, are didsplayed and can be maintained. Use the *Delete on Submit of Main Form* to delete a record that is no longer required. The record is deleted only after submitting the *Glossary Maintenance* main form.

Text Usage/Where Used — When clicked, will display where the glossary item is used. That may be a screen, a field property or a menu item. The 'Where Used' feature can be turned off using the 'Track Glossary Usage' option in System and Global Parameters.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3



Users are assigned to the Default Glossary by default.

They can be assigned to another defined Glossary in which case the user will see the translated text from the “Converted to” column.

Buttons

Submit	Update all amended glossary items and delete records flagged as Delete on Submit.
Cancel	Clear the form. Any changes made are not updated.

Chapter 16 – Entity Based Security and Lockdown

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Entity Definition

To enable more sophisticated security we have introduced a concept of Entity Based Security [EBS].

This enables any field in any file to become a security point.

Under EBS it is a simple process to create a security profile for a user that allows/disallows access based on a value in a file.

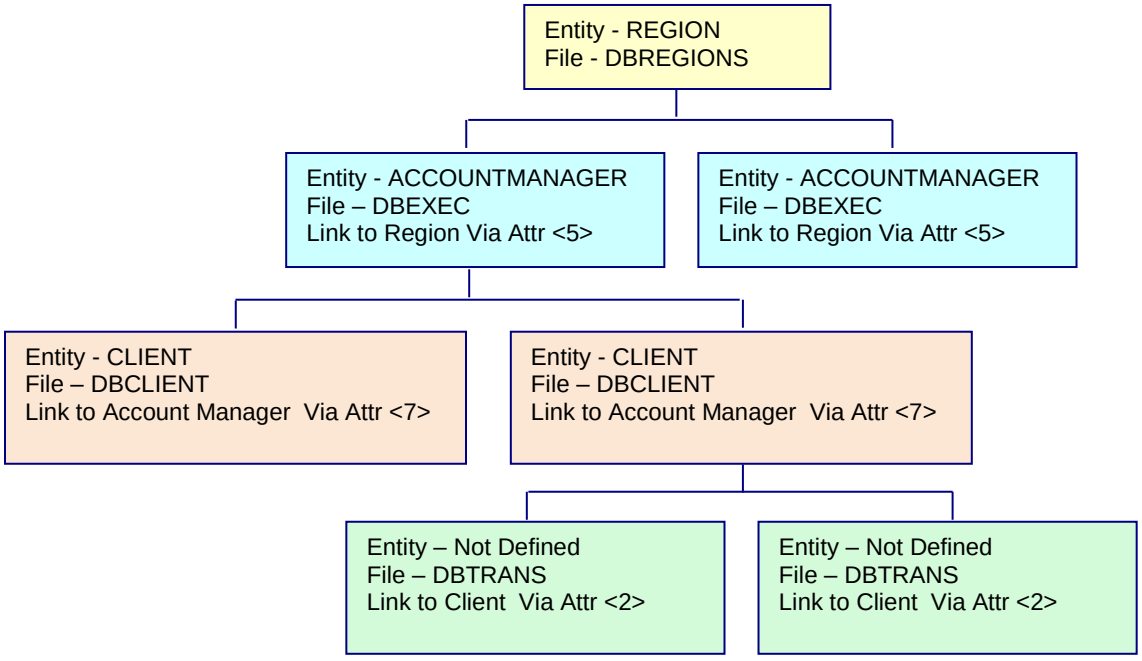
Eg: User Group EasternRegion are allowed update access to all Account Managers that belong to Eastern Region. They are also restricted from viewing any Account Managers in other regions.

Every client record contains an Account Manager field.

As a result of the relationship between Account Manager and Client, Users belonging to the EasternRegion Group are only provided with access to client records that have Account Managers that belong to EasternRegion.

Transaction records also contain client codes and therefore EasternRegion Users will only be able to access transactions for RegionEast.

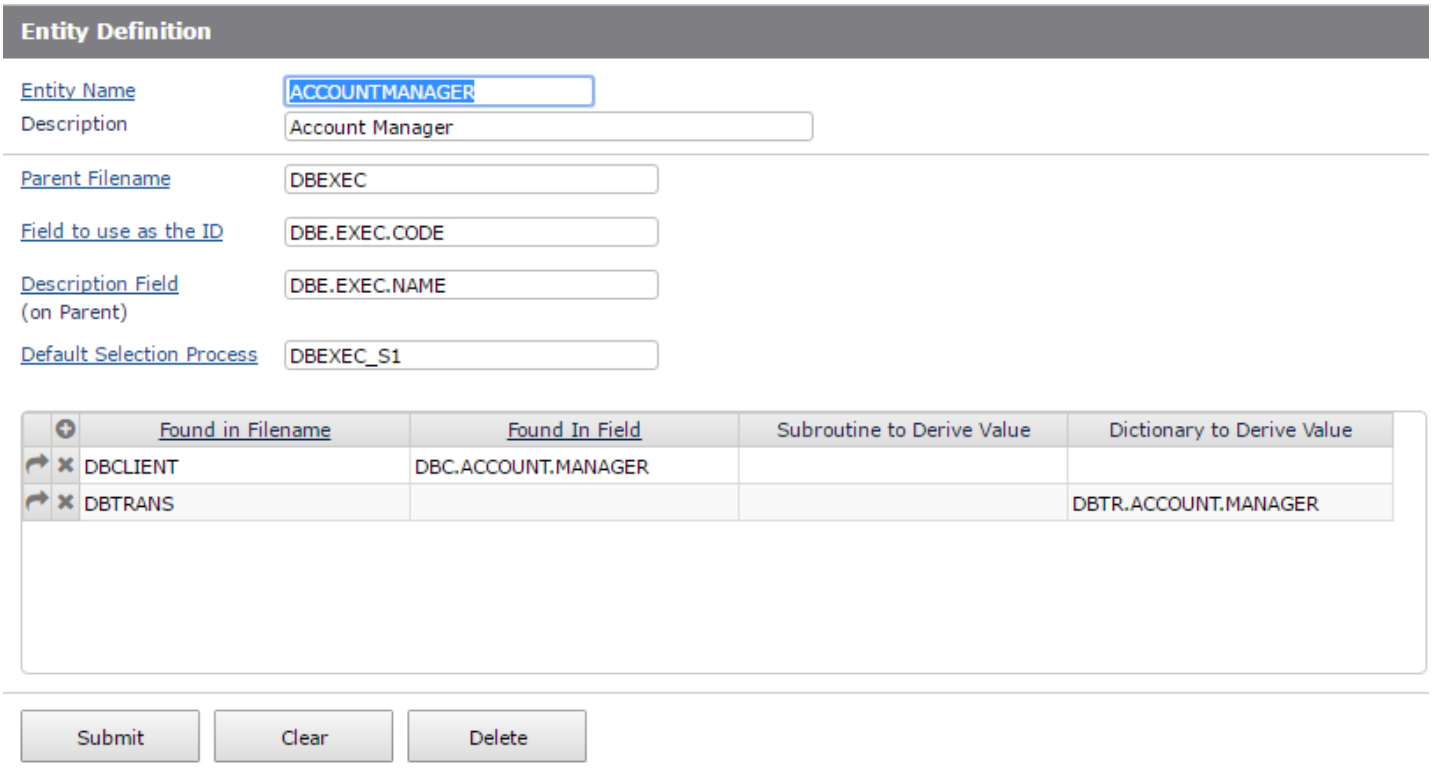
Example of an Entity Relationship model for the above entity structure



As most data in applications interrelate, controlling access via EBS is a very simple process.

In order for Entity Based Security to function correctly, these relationships must be defined.

The Entity Definition Form is used to define these relationships.



Prompts

Entity Name	This field is used to define the name of the entity for the entire application. For the Entity Security, this name will be used to control access of entity members by end users.
Description	Enter the description for the entity
Parent Filename	This field defines the name of the file that contains the parent record for the entity being named. In the example above file DBEXEC contains the reference details for account managers and thus is the parent file. Another example is a client file (DBCLIENT) would contain the parent record for the entity of Customer, whereas a transaction file (DBTRANS), although it too contains the client code, is not the parent file.
Field to use as the ID	This field is used to determine the name of the field that is to be used to derive the key both at a DesignBais and at a dictionary level. Typically this field would be assigned an attribute 0 with no other modifiers.
Description Field (on Parent)	This field contain the value of the DesignBais Field Property that contains an attribute reference to the description of the entity member. In our example the account manager name is in a field called DBE.EXEC.NAME. For a CLIENT entity with a parent filename of DBCLIENT the DesignBais field property entry of DBC.CLIENT.NAME contains the attribute where the description for the client resides.
Default Selection Process	In order to make the identification of Entity Member Codes easier for the end user, it is recommended that you assign a default search for each entity.

Found In Filename	This multivalue field is used to define the filenames that the entity resides in, where the entity is not the parent record. Eg. The DBTRANS file contains a reference to a Client Code (DBT.CLIENT.CODE) thus establishing an indirect link to the account manager.
Found In Field	This field contains the name of the DesignBais Field Property item that references the named entity in the parent file.
Subroutine to Derive Value	There may be instances where the member value of an entity must be calculated by a subroutine. If this is the case the Subroutine must be named in this field. The Subroutine must return the member value in the common variable DBVALUE
Dictionary to Derive Value	In some instances, it may be necessary to derive the value of the entity member value within a correlative or I-TYPE. If so, enter the name of the dictionary definition into this field.

Buttons

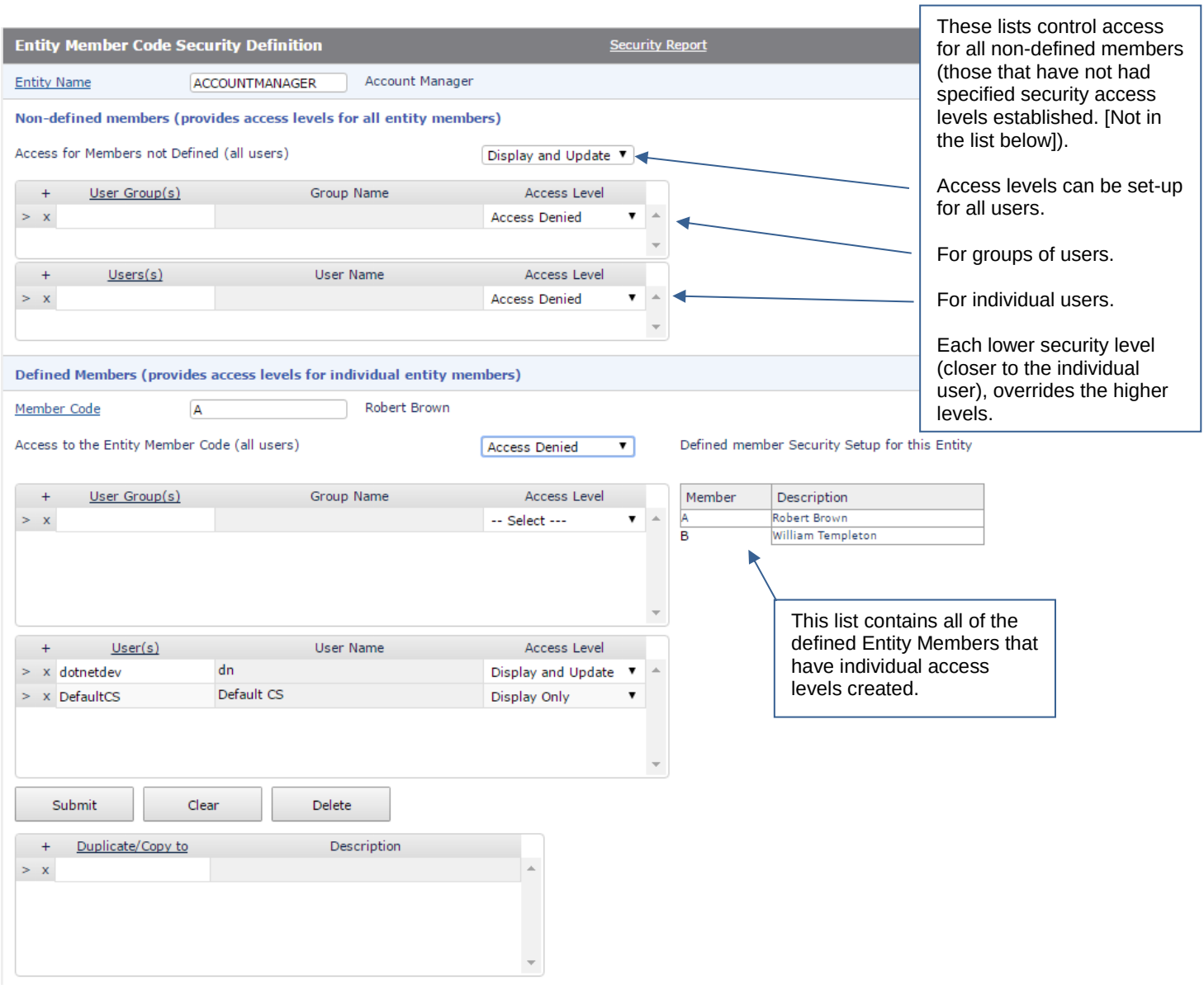
Submit	Updates the Entity Record. This Entity Definition Record is stored on the DBIPROP and DBIFILES files. These files need to be distributed with your application to make security function correctly. An Entity Name of “EXAMPLE.NAME” will create a record on DBIPROP with an id of “DBSENTITY*EXAMPLE.NAME”. This record will hold the details of all fields that are defined in the “Found in Filename / Found in Field” lists. The DBIPROP record “DBSENTITY.LIST” holds the list of all created Entity Definitions. The DBIFILES record for the Parent Filename file is also updated with the details of the fields defined in all Entity Definitions for the Parent Filename. These are in the fields: DBIFI.ENTITY.NAME DBIFI.ENTITY.POS DBIFI.ENTITY.DERIVED.IN.SUBR DBIFI.ENTITY.DERIVED.IN.DICT DBIFI.ENTITY.FIELD
Clear	Clears the form without updating.
Delete	Removes the Entity Definition Record

Entity Security

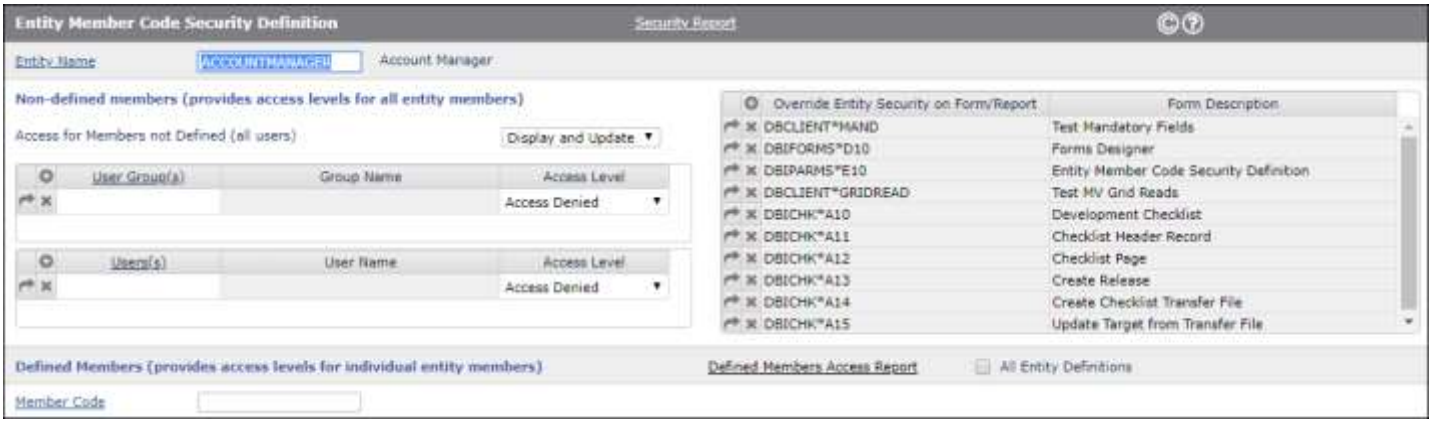
The Entity Security form is directed towards the End-user Administrator. Users of this calibre can create access profiles for entity members, users and groups.

An Entity Member record is a valid record that belongs to each Parent File. For example; a client file has a client record and a client identifier (attribute 0). The member code is the client identifier and the member record is the client record.

This form is used to define the Entity Members (for each Entity) that a user or user group can access.



From Release 8 onwards the list of forms in which entity security is to be overridden is displayed for reference on the Entity Security maintenance form.

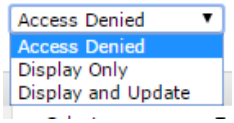


Prompts

Entity Name — This field is used to define the name of the entity for the entire application. For the Entity Security, this name will be used to control access of entity members by end users.

Member Code — An Entity Member code is the identifier for the Entity Record.

Access to the Entity Member Code (all users)



This controls access to the Entity Member Code for everyone.

All Access Level Fields have three possible selections.

1. Access Denied. — No access is granted to the Entity Member record, or any record that contains the Entity Member Code

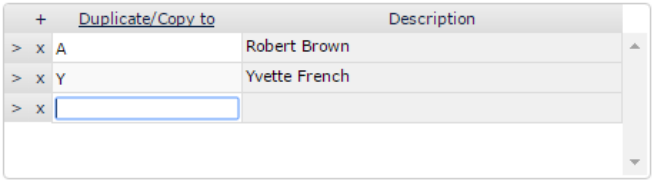
2. Display Only — Display Only access is granted to the Entity Member record, or any record that contains the Entity Member code.

3. Display and Update — Display Only access is granted to the Entity Member record, or any record that contains the Entity Member code.

User Group(s) — Used to define the list of user groups that have some form of access control for the entity member. Access levels specified at the group level override any access granted or denied at the Everyone level.

User(s) — Used to define the list of users that have some form of access control for the entity member. Access levels specified at the user level override any access granted or denied at the Everyone and User Group levels.

Duplicate/Copy to — This field is used to copy a saved Member's definition to other Member records. Simply add the list of members to this Multivalue field and press the Submit button.



Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Buttons

Submit	Updates the Entity Member access levels and performs any Duplicate copies as defined.
Clear	Clears the form
Delete	Deletes the current Member Entity Record

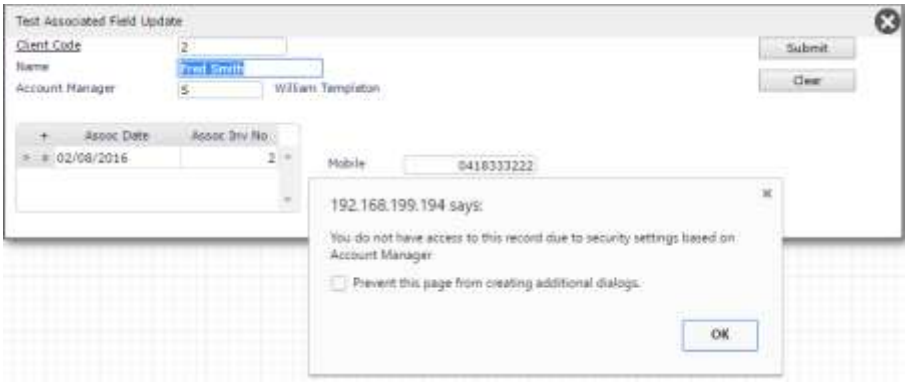
The Entity Security Definition is held on DBIPARMS in 3 records DBSENTITY, DBSENTITY.DEFAULT and DBSENTITYOVERRIDE.

Entity Security Processing

The following examples demonstrate how Entity Security is invoked. Consider the form below where the record for Client Code 2 is displayed. The Account Manager code is ‘B’ and under the Account Manager Entity Security setup the user is permitted to view records for Account Manager ‘B’.

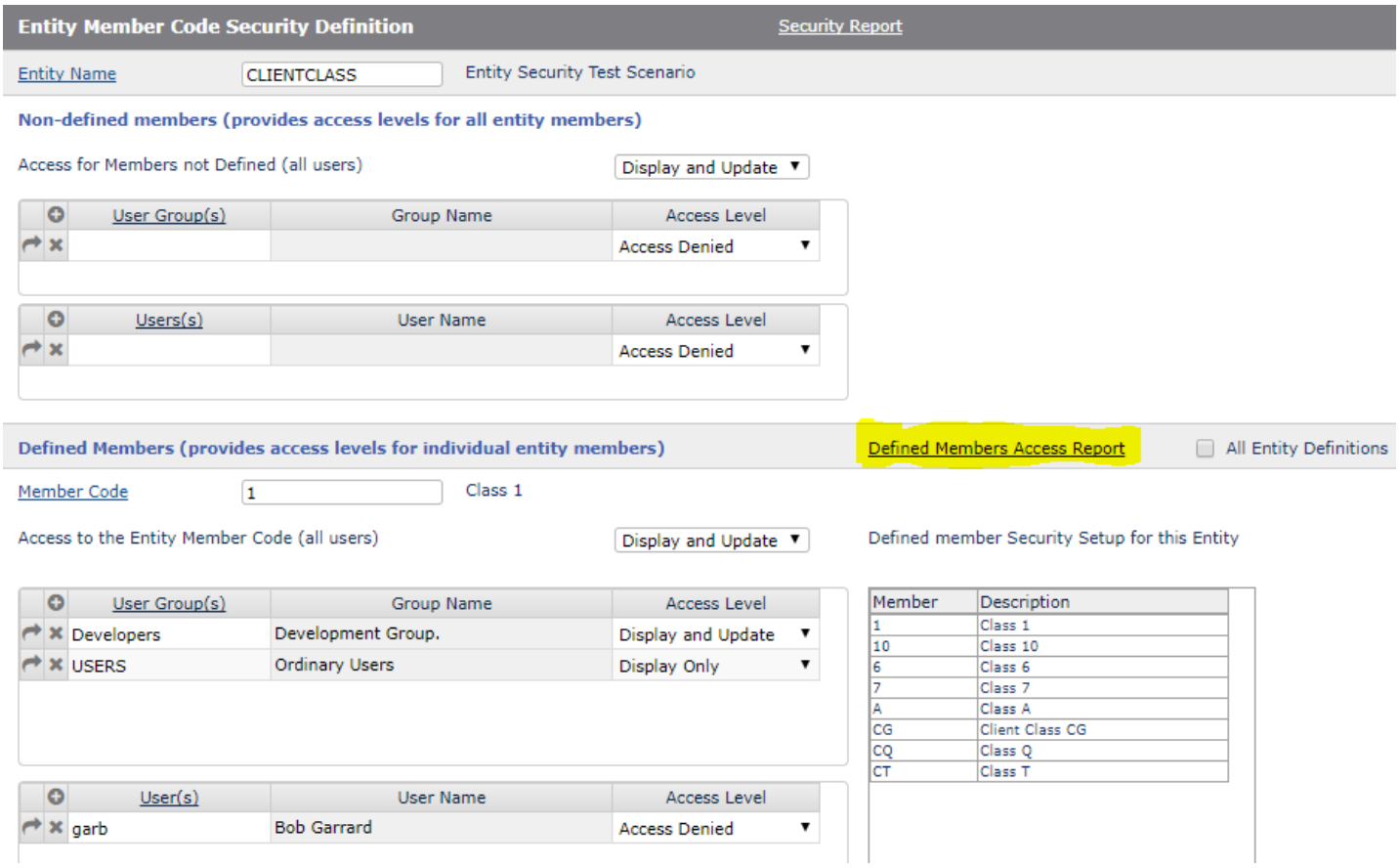
If the user enters the code for a client with an Account Manager code that the user is not permitted to display then a message is displayed and the user cannot access the record.

If the user attempts to set the Account Manager code to a value that is not permitted under the Entity Security setup then the following message is displayed and the change is not allowed.



Defined Members Access Report

This report combines all the entity member group and user access levels for all defined members (Record Ids) within the Entity.The check box “All Entity Definitions” allows the report to include all Entity Security definitions rather than just the currently displayed definition.



The report can be sorted on any column thus allowing the user to see the access levels for a Group or a User for all member records. Currently the maximum allowed rows for this report is 1800 to ensure that the browser is able to process the report.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Entity Security Override

There may be some forms in your application that you require to bypass the Entity Security processes.

This form allows you to identify those forms. Note that changes to the exclusion list will not take effect until the browser is refreshed.

Care is needed in adding forms to this list. If a set of forms is called as Sub Forms and one or more of these forms are in this list then all the Sub Forms should be in the list. As an example of the problem that may arise consider that the calling form and the Sub Forms may display a dropdown list of entities where these entities are effected by security. If entity security causes the dropdown list to contract, then the position marker of the selected item may point to the wrong entity if only one of the Sub Forms is in the Override list, since the list differs between forms based on entity security.

Prompts

Override Entity Security on Form/Report

This field is used to define the forms or reports that are exempt from the Entity Based Security Processes.

Buttons

Submit — Updates the list of form/reports.

Lockdown Mode

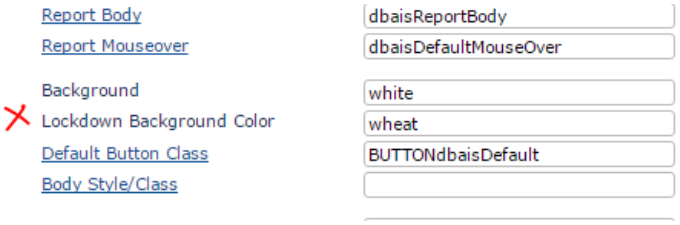
Lockdown Mode allows the End-User Administrator to apply security setting on any input field, button or hyperlink on the form. Lockdown Mode can provide both Field and Function based security on a form.

Lockdown cannot be applied to Multivalue grid fields.

Press the “Turn Lockdown Mode On” button to turn lockdown mode on for the session.

Execute the form that is to have security applied to it. Either use the ‘Run a Form’ option or run the form from a menu.

Any form loaded whilst in lockdown mode will have a different colored background. This is to indicate to the user that they are in Lockdown mode. The color is defined in the Style Group and if no color is defined then the default is “wheat”.



Locking down a button or a field requires a few steps to be completed as shown in the following example.

Example Lockdown Process

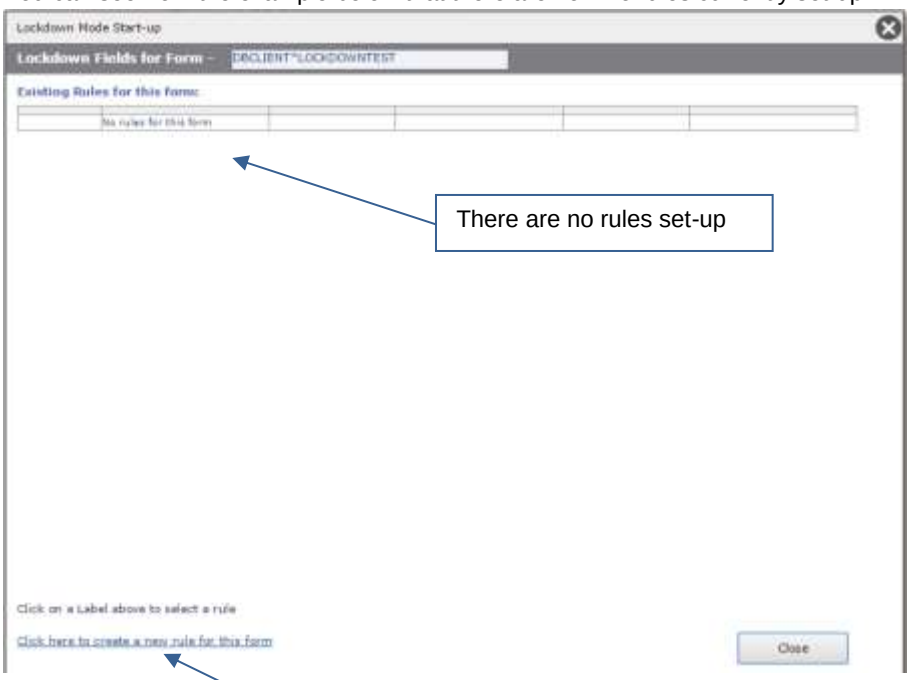
In this example our End-User Administrator requires that the Client Maintenance form have the Submit and Delete Buttons disabled for any client that has a Credit Stop flag. All “Ordinary Users” will be prohibited from pressing the Submit or Delete buttons in this case.

Step 1. **Turn Lockdown Mode on**

Step 2. **Run the required form (You will notice that the background color is different).**

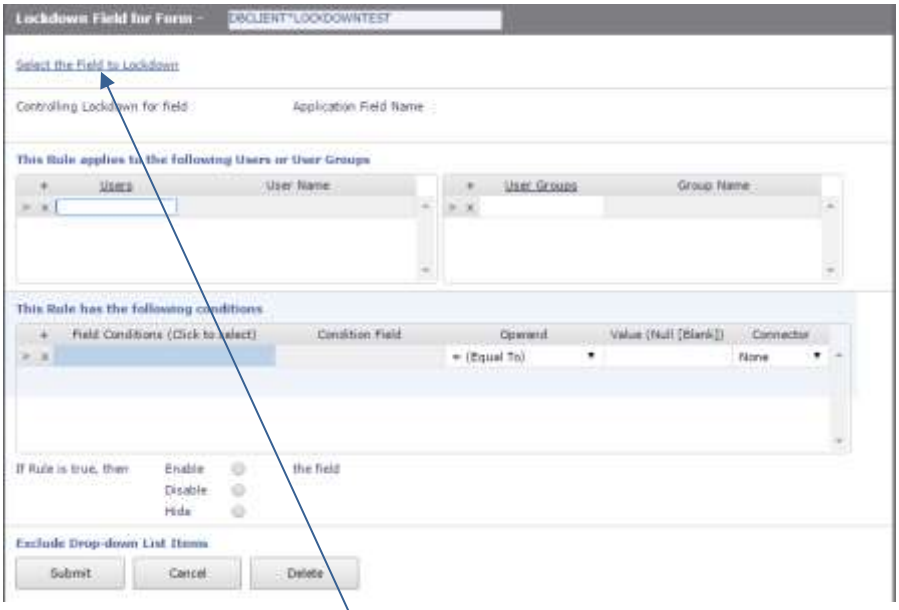
Step 3. **Set the Disable condition for the Submit Button**

Move the mouse across any button or input field on the form and Press control+click (click = left mouse button)
The following form will be displayed
You can see from the example below that there are now no rules currently set-up.



Step 4. **Click on the "Click here to create a new rule for this form" hyperlink**

The following form will be displayed

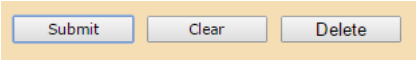


Step 5. **You must now select the field to lockdown.**

Click on the "Select the Field to Lockdown" hyperlink. This will return the user to the original form to select

Step 6. **You are now presented with the original form**.

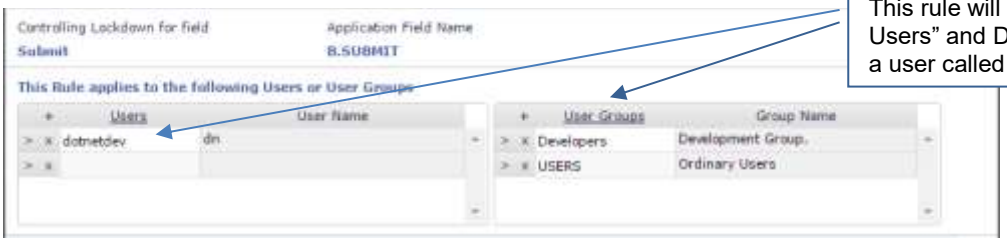
Move the mouse pointer over the Submit button and press Control+Click



The Lockdown rule set form is displayed again.

The Submit button, B.SUBMIT is now selected to have a rule applied to it

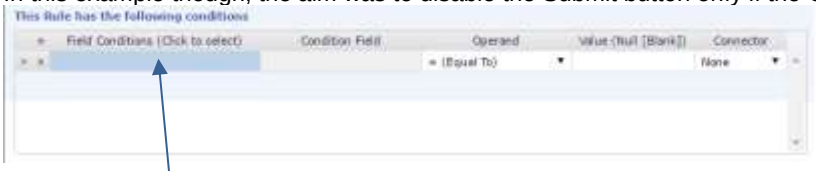
Step 7. **Select the Users and/or User Groups that this condition applies to:**



This rule will apply to "Ordinary Users" and Developers, and to a user called "dotnetdev".

Step 8. **Select the Field or Fields that will determine if rule.**
You may leave this section blank. If so the rule would apply to any user of the "Ordinary Users" group. This is useful if you wish to disable the Submit button for users in a group (or individual users) regardless of the data on the form.

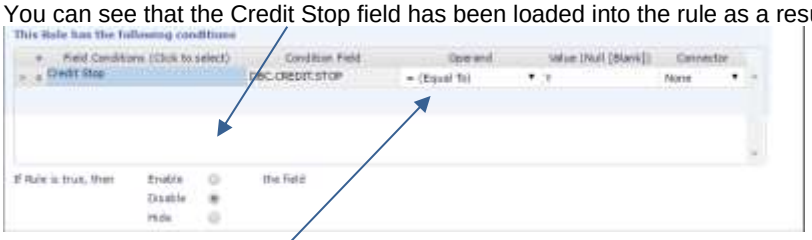
In this example though, the aim was to disable the Submit button only if the Credit Stop flag was true.



Click on the **shaded** area to select the field on the Client Maintenance form that will drive this rule.

Step 9. **The Client Maintenance form will be re-displayed.**
Position the mouse on the *Credit Stop Check Box* and press control+click

Step 10. **Now we can complete the rule**
You can see that the Credit Stop field has been loaded into the rule as a result of Step 9.



Set the Operand to "= (Equal to)" as we want this rule to enact when the Credit Stop Flag is a "Y".

Set the Connector to none. If we have multiple conditions to enact a rule then we can choose the Or, And or Next Field connectors.

Examples of Connector Usage

Or	If	"Credit Stop"	=	Y	**Or**
		"Account Status"	=	S	

If the "Credit Stop" field is equal to a "Y" **or** the "Account Status" = "S" then the rule is true.

And	If	"Credit Stop"	=	Y	**And**
		"Account Status"	=	S	

If the "Credit Stop" field is equal to a "Y" **and** the "Account Status" = "S" then the rule is true.

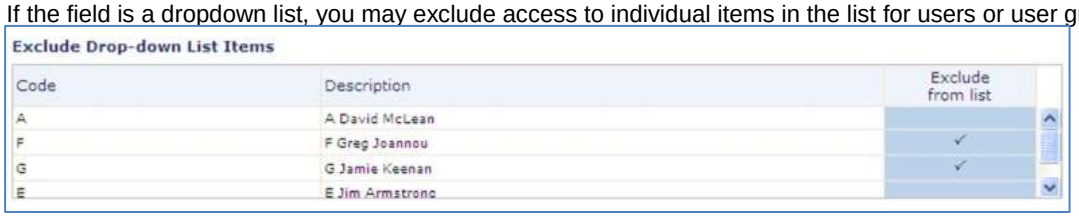
Next Field	If	"Credit Stop"	=	**Next Field**
		"Account Status"		

If the "Credit Stop" field is equal to the "Account Status" (the next field) then the rule is true.

Step 11. **Select what to do with the field that we are locking down.**
In this case the Submit (B.SUBMIT) button.

If the rule is true then the Submit button is Disabled.

If the field is a dropdown list, you may exclude access to individual items in the list for users or user groups.



This is achieved by clicking in the "Exclude from list" column. A tick will be displayed to denote the exclusion.

Step 12. **Submit the Rule**
Press the Submit button to save the rule.

There is now a single rule for the form DBCLIENT*LOCKDOWNTEST

This rule applies to the Submit (B.SUBMIT) button.

Step 13. **Now repeat the steps for the Delete Button.**

There are now two rules: One for the Submit button and one for the Delete button.

Lockdown Fields for Form - DBCLIENT*LOCKDOWNTEST

Existing Rules for this form:

Button	**Delete**				
App. Name	**B.DELETE**	**Applies to Users**		**Applies to Groups**	
		dotnetdev	dn		
	Field to Test	**Application Label**	**Operand**	**Value**	**Connector**
Applies If	Credit Stop	DBC.CREDIT.STOP	=	Y	
Then	**Hide the Button**				
	List elements excluded				
Button	**Submit**				
App. Name	**B.SUBMIT**	**Applies to Users**		**Applies to Groups**	
		dotnetdev	dn	Developers	Development Group.
				USERS	Ordinary Users
	Field to Test	**Application Label**	**Operand**	**Value**	**Connector**
Applies If	Credit Stop	DBC.CREDIT.STOP	=	Y	And
Then	**Hide the Button**				
	List elements excluded				

Step 14. **Test the results.**
It is always best to start a new window to test the results. The reason for this is that Lockdown and other display characteristics are set-up at form load in an attempt to maintain high performance levels.

Both the Submit and the Delete buttons are hidden when the Credit Stop flag is “Y” for users in the USERS or Developers group or for user id ‘dotnetdev’.

If the rules are to disable the buttons then the form will appear like this.

This is a very useful tool to avoid application development work when field-based security is required, particularly if the security is based on functional results.

Spell Checking

DesignBais no longer provides spell checking as it relies on the spell checking ability built into modern browsers.

Chapter 17 – File Upload Process

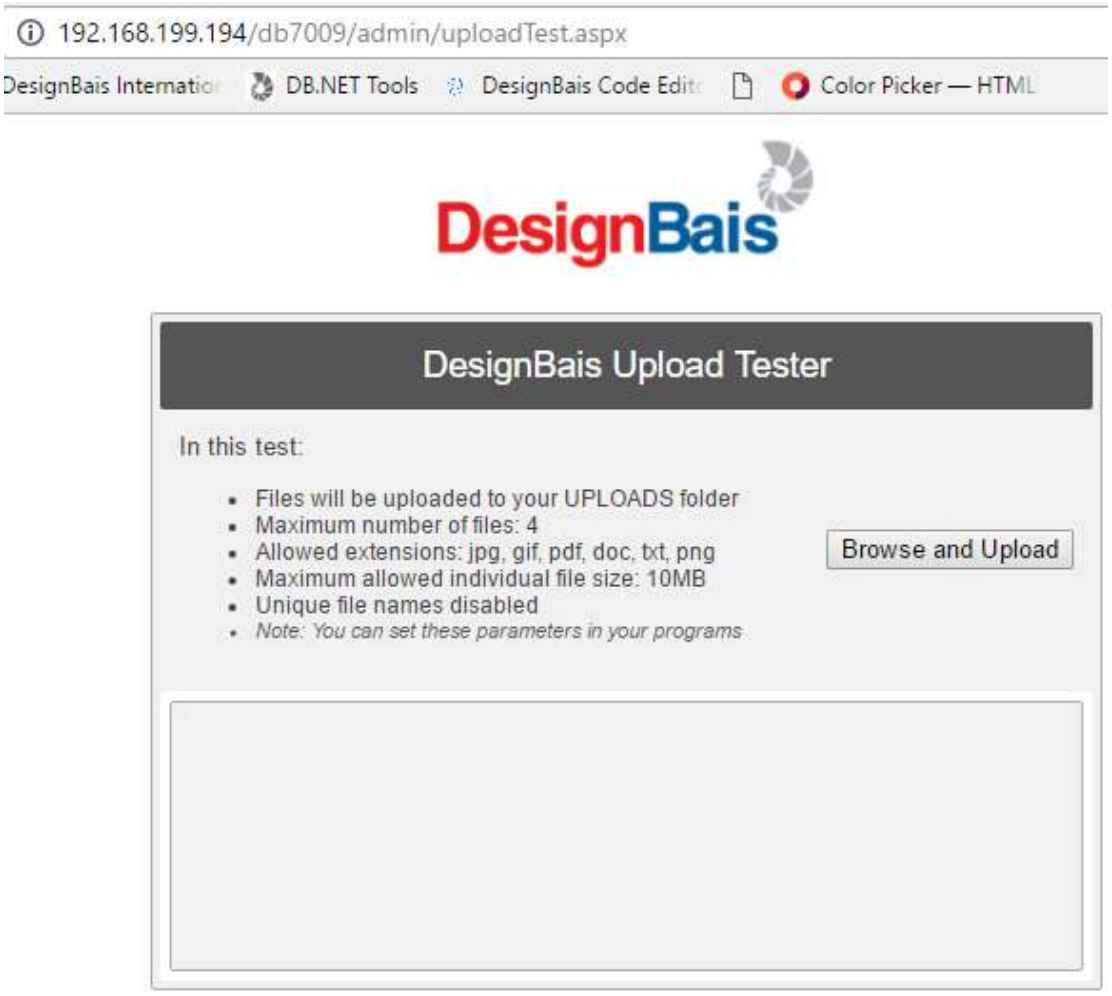
File Upload Process

The file upload process is a simple set of program functions that instruct the DesignBais engine to invoke a file upload facility.

This will allow files of nominated types to be selectively copied from the client computer to the webserver.

This facility also provides the file details back to the calling programs to allow the developer to attach the file name to a record for saving.

There is an upload tester available at “admin/uploadTest.aspx” which can be used to test the upload process:



The following text details step-by-step instructions on how to invoke the file upload process from your Basic subroutine:

Step 1.

An event from a form is raised in response to a client action. In this example the event is raised from a button.

```
CASE EVENTSOURCE = "B.UPLOAD" AND SCREEN.NO[1,6] = "UPLOAD"
   FTYPS = ";.txt.jpg.gif.pdf.docx.xmlx"
   PATH = ";UPLOADS/":DBIACCOUNT:"/myfiles"
   MAXF = ";10"
   MAXS = ";100"
   IF SCREEN.NO = "UPLOAD2" THEN
      UFN = ";false"
   END ELSE
      UFN = ";true"
   END
   URL = SESSION.ID:";uploadWindow":PATH:FTYPS:MAXF:MAXS:UFN
   DBUPLOADRETURNPROCESS<1,1> = "DB.I.DBCLIENT"
   DBUPLOADRETURNPROCESS<1,2> = "R.UPLOAD"
   DBCALLURL = URL
   DBWORK<DBC.DISPLAY1.WK>=URL
```

DBCALLURL components are defined as follows and must be separated by a semi-colon (;) :

- SESSION.ID DesignBais session id variable
- uploadWindow Invoke the upload window
- PATH This path is the relative path from the web address.
  In the above case, the path is
  UPLOADS/*AccountName/myfiles*

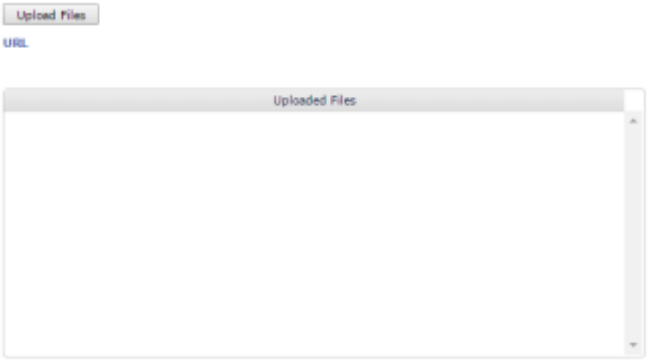
  In this example the files uploaded would be stored in a directory with the same name as the account in which the upload is run. If the directory does not exist, it will be created.

  The path can include a DesignBais variable such as DBKEY, in which case files can be stored in a directory, the name of which includes the current value in DBKEY.
- FTYPS The types of files that can be uploaded separated by a period(.)
- MAXF The maximum number of files that can be uploaded.
- MAXS The maximum total size of the files that can be uploaded in mb.
- UFN 'true' = filenames will be replaced with unique keys. Original file name is not sent by the webcomp
  'false' = retains filenames as selected. The file is replaced if it already exists.

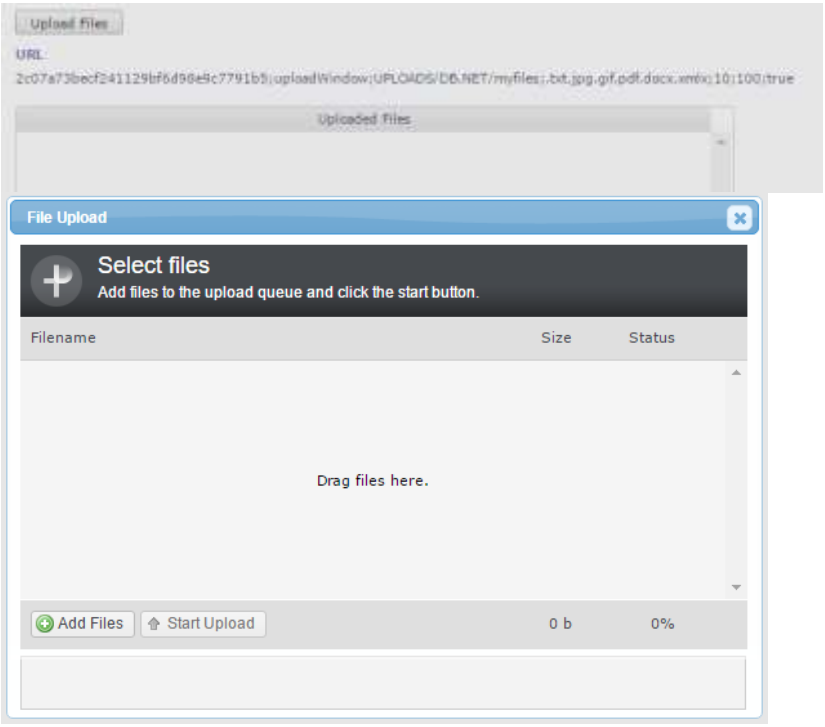
Step 2.

The variable DBUPLOADRETURNPROCESS defines the name of the program and the name of the eventsource to which control is passed on completion of the upload. Value 1 holds the program name, value 2 holds the eventsource name.

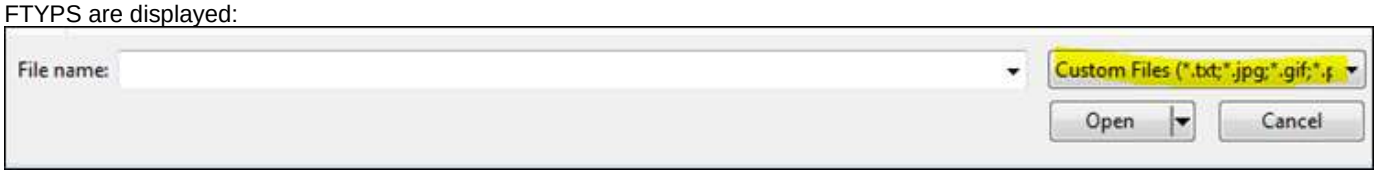
Here is a simple example of the Upload process.



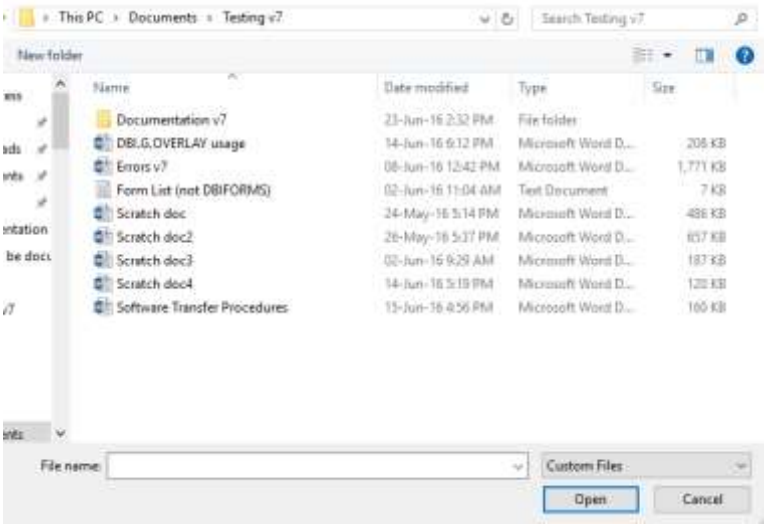
Clicking the Upload Files button executes the code shown above in Step 1. The URL that is sent is displayed.

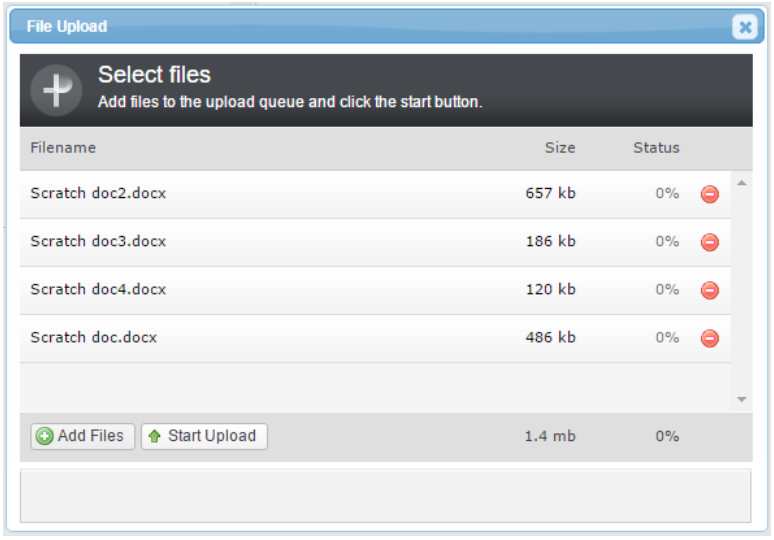


Click the Add Files button. Browse to the directory from which files are to be uploaded. Only files of types designated in FTYPS are displayed:

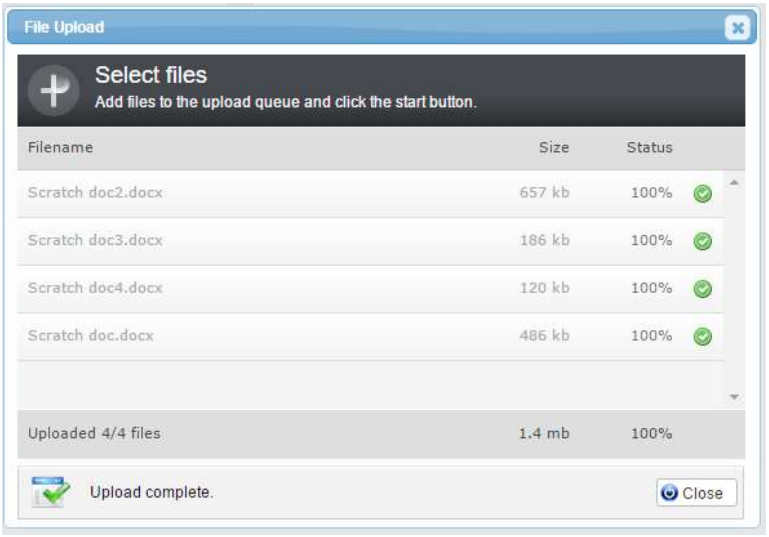


You can either highlight a file and click Open, or you can drag the file into the “Drag files here” window.

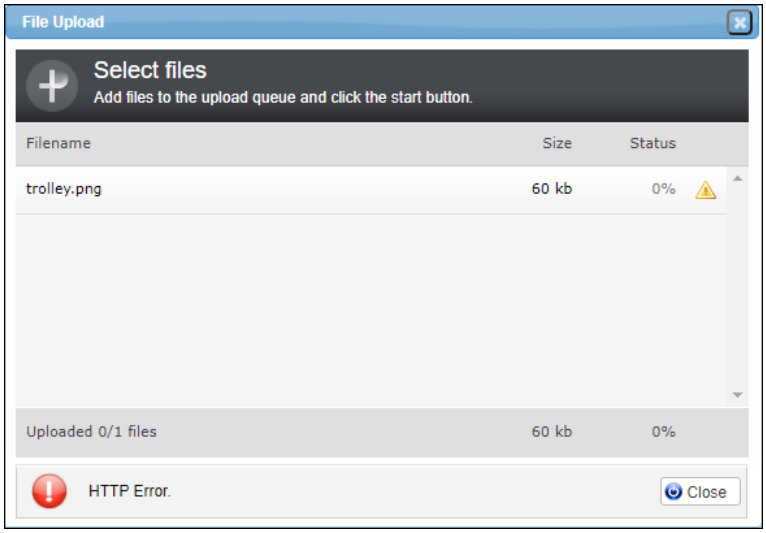




When all required files have been selected click Start Upload, then click Close.



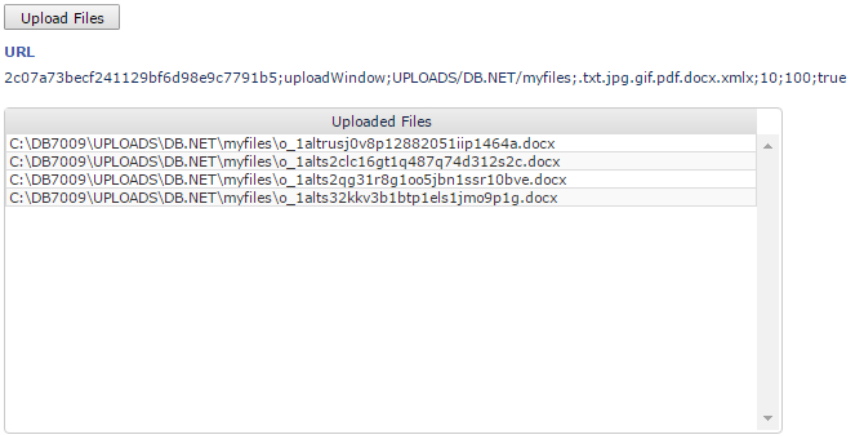
If a file cannot be found the File Upload will display an error at the base of the form:



Control is then passed backed to the program named in DBUPLOADRETURNPROCESS and, in our example below, the code within the UPLOAD event is executed, with the defined eventsource.

```
*
UPLOAD:
*
    BEGIN CASE
       CASE EVENTSOURCE = "R.UPLOAD"
          UFS=PROCESS.PARAMETER
          *DBDS<-1> = UFS
          UFMAX=DCOUNT(UFS,VM)
          FOR ZZ=1 TO UFMAX
             UPFILE=UFS<1,ZZ,1>
             DBWORK<DEM.DISPLAY2.WK,ZZ>=UPFILE
             DBWORK<DEM.DISPLAY3.WK,ZZ>=UFS<1,ZZ,2>
          NEXT ZZ
    END CASE
    RETURN
```

This code is for demonstration purposes. It merely lists the names of the uploaded files. Note that the file names are returned in PROCESS.PARAMETER, value mark, then subvalue mark, delimited.



Note:
If the uploads folder is needed on a remote machine (rather than the web server) then the following three default folders under a DBNET should be deleted and new IIS virtual folders should be created pointing to the remote machine folders using unc paths (e.g. \\192.168.199.118\somefolder) :

- uploads

- uploadsin
- uplprogress

There are three uploads mechanisms in DesignBais:

- Normal upload (e.g. our developer tools demo upload)
- Webservice upload (uses wstransfer.ashx)
- RD upload

web.config and IIS MIME types (Multipurpose Internet Mail Extensions) should be set for all of the above.

web.config:

```
<httpRuntime targetFramework="4.5" maxRequestLength="2000000000" requestValidationMode="2.0"
executionTimeout="300"/>
```

maxRequestLength is in bytes and executionTimeout is in seconds

IIS MIME Types (rarely needs a change):

MIME types must be added to IIS if missing (most extensions don't miss; e.g. .mov is already there)

Allowed Extensions:

Normal Uploads	Nothing is needed in admin/db.config. The DATACOMP determines the allowed extensions.
Webservice Uploads	Use wsUpload/extensions node in admin/db.config
RD Uploads	Use RDallowedUploadExtensions node in admin/db.config

Allowed file size:

Normal Uploads	Nothing is needed in admin/db.config. The application code determines the allowed file size. maxRequestLength takes effect if it is smaller than what DATACOMP requires.
Webservice Uploads	Nothing is needed in admin/db.config. maxRequestLength is in effect.
RD Uploads	Configure RDallowedUploadSizeMB node in admin/db.config. The smaller of RDallowedUploadSizeMB and maxRequestLength takes effect.

Refer to the DesignBais Responsive Design manual for more details. Note that these db.config parameters only apply in the Designer. This means, in RD designer, when you add a new FILE-UPLOAD element to the form, that element's attributes are set based on those parameters found in db.config. Once the page is published, the parameters found in db.config do not have any effect. Changing those parameters in db.config won't make any difference in run time. You must use rdSetAttribute() to set these parameters as needed at run time!

Chapter 18 – Web Service

Invoking a Web Service

A new common variable DBCALLWS is used to invoke a web service.

This is a multi-attributed variable as defined below:

DBCALLWS<1>	Header Names MV delimited
DBCALLWS<2>	Header Values MV delimited
DBCALLWS<3>	Url to invoke
DBCALLWS<4>	Data input for web service
DBCALLWS<5>	Login id
DBCALLWS<6>	Password
DBCALLWS<7>	ProxyUrl
DBCALLWS<8>	ProxyPort
* Web Client Authentication	
DBCALLWS<9>	RemoveBasic - "yes" will remove "Basic:" from authentication header
DBCALLWS<10>	IgnoreSSLErrors - "yes" will ignore errors
DBCALLWS<11>	Method = defaults to POST can be GET
* Result Handler	
DBCALLWS<12>	Program to invoke on return
DBCALLWS<13>	Program parameter
* For DesignBais Response handling	
DBCALLWS<14>	Response passed to PROCESS.PARAMETER
* REST	
DBCALLWS<15>	Debug Indicator
DBCALLWS<16>	Timeout
DBCALLWS<17>	Method – mandatory used to indicate REST
DBCALLWS<18>	Relative Path if method is UPLOAD
DBCALLWS<19>	clientCert
DBCALLWS<20>	clientcertPass

NB: Invoking a RESTful web service will not use positions 9 to 11.

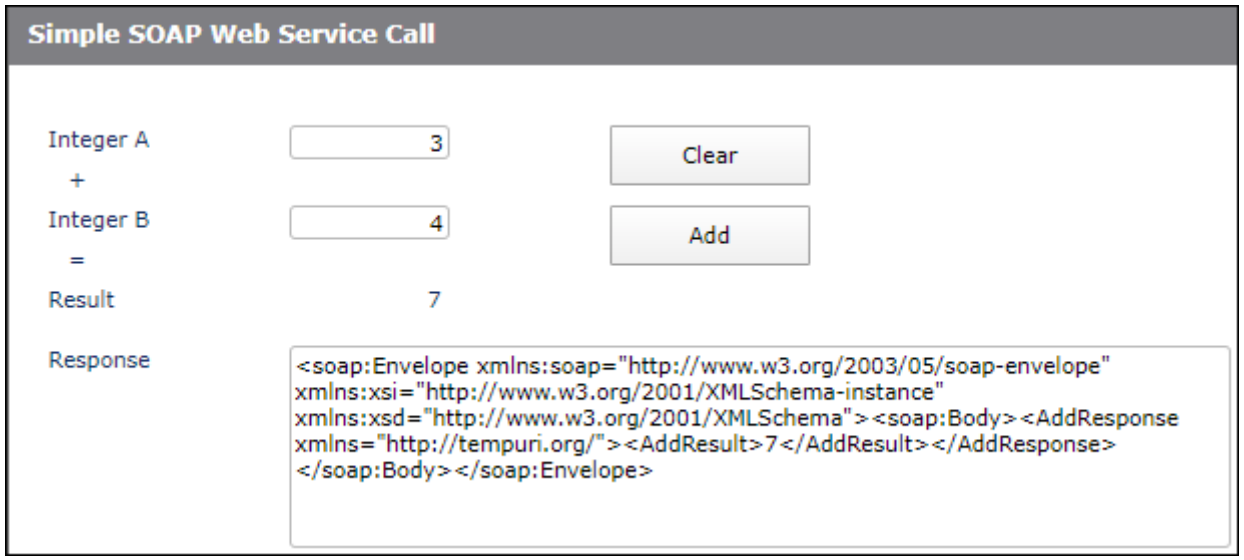
Rest XML Package:

<rest> : Parent node. Can take 3 attributes txid: 16-character alphanumeric transaction ID debug: Default value set to false. Activates debug logging if set to “true”. timeout: Timeout value in seconds. Kills process if doesn’t receive a response.	Mandatory
<targeturl>: External service URL address. Must be in <![CDATA[]]> format.	Mandatory
<targetmethod>: HTTP method to be used.	Mandatory
<proxyurl>: Proxy URL address to hit. Must be in <![CDATA["URL"]]> form.	Optional
<proxyport>: Proxy Port number.	Optional
<data>: Data to send to external service. Must be encoded in Base64 format.	Optional
<filepath>: File path of the file to upload. Mandatory if HTTP method is ‘UPLOAD’.	Optional
<header>: Header declaration for HTTP request. Must be in <![CDATA[header:value]]> format.	Optional
<login>: Webservice credential login. Must be <!CDATA[]]> format.	Optional
<password>: Webservice credential password. Must be <!CDATA[]]> format.	Optional
<clientcert>: Must be in <![CDATA["certificate name"]]> form.	Optional
<clientcertPass>: Must be in <![CDATA["password"]]> form.	Optional

The digital security certificate must be located in **CertificatesDB** folder.

SOAP Web Service Example

Here is an example of the code used to call the Demo Forms Simple SOAP Web Service Call to add two integers.
In this example the call to the web service is initiated from a button called “B.ADD” in the form DBDEMO*SOAPADD.



This is the code for the *Add* button in the BUTTON event:

```
CASE EVENTSOURCE = "B.ADD"
   OP1 = FIELD(DBWORK<DEM.WORK2.WK>,".",1)
   OP1 = OCONV(OP1,"MCN") + 0
   OP2 = FIELD(DBWORK<DEM.WORK3.WK>,".",1)
   OP2 = OCONV(OP2,"MCN") + 0
   IF OP1 # DBWORK<DEM.WORK2.WK> OR OP2 # DBWORK<DEM.WORK3.WK> THEN
      DBPASS.DBVALUE.TO = 'DEM.WORK2.WK':VM:'DEM.WORK3.WK':VM:'DEM.WORK.DISP.WK'
      DBPASS.DBVALUE    = OP1:VM:OP2:VM:""
      DBBUTTONCLICK = 'B.ADD'
      RETURN
   END
   * Web Service Call Data
   DBCALLWS       = "Content-Type"                                    ;* <1> = Header Names  - MV delimited
   DBCALLWS<2>    = "application/soap+xml; charset=utf-8"             ;* <2> = Header Values - MV delimited
   DBCALLWS<3>    = "http://www.dneonline.com/calculator.asmx"        ;* wsUrl to invoke
   * wsData input for Web Service
   * *** Do NOT include "<?xml version='1.0' encoding='utf-8'?>"
   DBCALLWS<4>    = "<soap12:Envelope xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xmlns:xsd='http://www.w3.org/2001/XMLSchema'"
   DBCALLWS<4>   := " xmlns:soap12='http://www.w3.org/2003/05/soap-envelope'>"
   DBCALLWS<4>   := "<soap12:Body>"
   DBCALLWS<4>   := "<Add xmlns='http://tempuri.org/'>"
   DBCALLWS<4>   := "<intA>":OP1:"</intA>"
   DBCALLWS<4>   := "<intB>":OP2:"</intB>"
   DBCALLWS<4>   := "</Add>"
   DBCALLWS<4>   := "</soap12:Body></soap12:Envelope>"
   DBCALLWS<5>    = ""                                                ;* wsLogin id
   DBCALLWS<6>    = ""                                                ;* wsPassword
   * Web Client Authentication
   DBCALLWS<7>    = ""                                                ;* wcProxyUrl
   DBCALLWS<8>    = ""                                                ;* wcProxyPort
   DBCALLWS<9>    = ""                          ;* wcRemoveBasic - "yes" will remove "Basic:" from authentication header
   DBCALLWS<10>   = ""                                                ;* wcIgnoreSSLErrors - "yes" will ignore errors
   DBCALLWS<11>   = ""                                                ;* wcMethod = defaults to POST can be GET
   * Result Handler
   DBCALLWS<12>   = "DBI.I.DEMO"                                      ;* Program to invoke on return
```

```
   DBCALLWS<13>   = "B.LOADADD"                                        ;* Program parameter
```

Following the call to the web service control is returned to the nominated routine with PROCESS.EVENT set to REMOTERETURN. PROCESS.EVENTSOURCE is set to DBCALLWS<13>. The data from the web service is returned in PROCESS.PARAMETER. The web service provider defines the pattern of the output from the web service.

```
CASE EVENTSOURCE = "B.LOADADD"
   * <soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   * <soap:Body>
   *    <AddResponse xmlns="http://tempuri.org/">
   *         <AddResult>8</AddResult>
   *    </AddResponse>
   * </soap:Body>
   * </soap:Envelope>
   SPOS = INDEX(PROCESS.PARAMETER,"<AddResult>",1) + 11
   EPOS = INDEX(PROCESS.PARAMETER,"</AddResult>",1)
   LENG = EPOS - SPOS
   RES  = PROCESS.PARAMETER[SPOS,LENG]
   DBWORK<DEM.WORK4.WK> = RES
   DBWORK<DEM.WORK.DISP.WK> = PROCESS.PARAMETER
```

Using DBTIMER to verify if a file exists

See Using DBTIMER to verify if a file exists

Providing a Web Service

The ability to provide a webservice is new in DesignBais Release 7/8. The special url that triggers the Web Service Subroutine is:

http://domain/DesignBaisnet/webservice.ashx?ac=x&stuff=y

The target account is determined by the querystring parameter "*ac*". The *?ac=string* is a qcode entryPoint in the db.config file, as explained in the Web Component Manual. This entry point is used to direct DesignBais to a specified data account.

Other data can also be passed via other custom querystring parameters.

The actual data (received from the caller) is an xml string. For XML and Soap it is always contained in a <dbWebService> node. For JSON the <dbWebService> is not required (see below).

<screenData>
<security…all attributes can be ignored…/>
<baAction>GETWEBSERVICE</baAction>
…
<dbWebService>
<x>new test</x>
</dbWebService>
</screenData>

Note that the external caller sends the red-colored segment, enclosed in the blue-colored <dbWebService></dbWebService> tags.

The DesignBais Data Component of DesignBais takes whatever is inside the blue tags as the input.

In response to the input XML string given above, the Data Component returns:

<webServiceReturn>
<![CDATA[<myData>abc</myData>]]>
</webServiceReturn>

The Web Component passes only the red segment to the caller. So, from the caller's point of view, the communications looks like this:

Caller sends:
<dbWebService>
<x>new test</x>
</dbWebService>

Caller receives:
<myData>abc</myData>

Note that if the web service returns a JSON response then it will look like this:
<wsReturn><![CDATA[JSON_RESPONSE]]></wsReturn>

whereas an XML response looks like this:
<wsReturn>XML_RESPONSE</wsReturn>

In case of an error, the caller receives from the Web Component:

<error><![CDATA[this is the error message]]></error>

WEBSERVICE calls do not create a session or seat count.

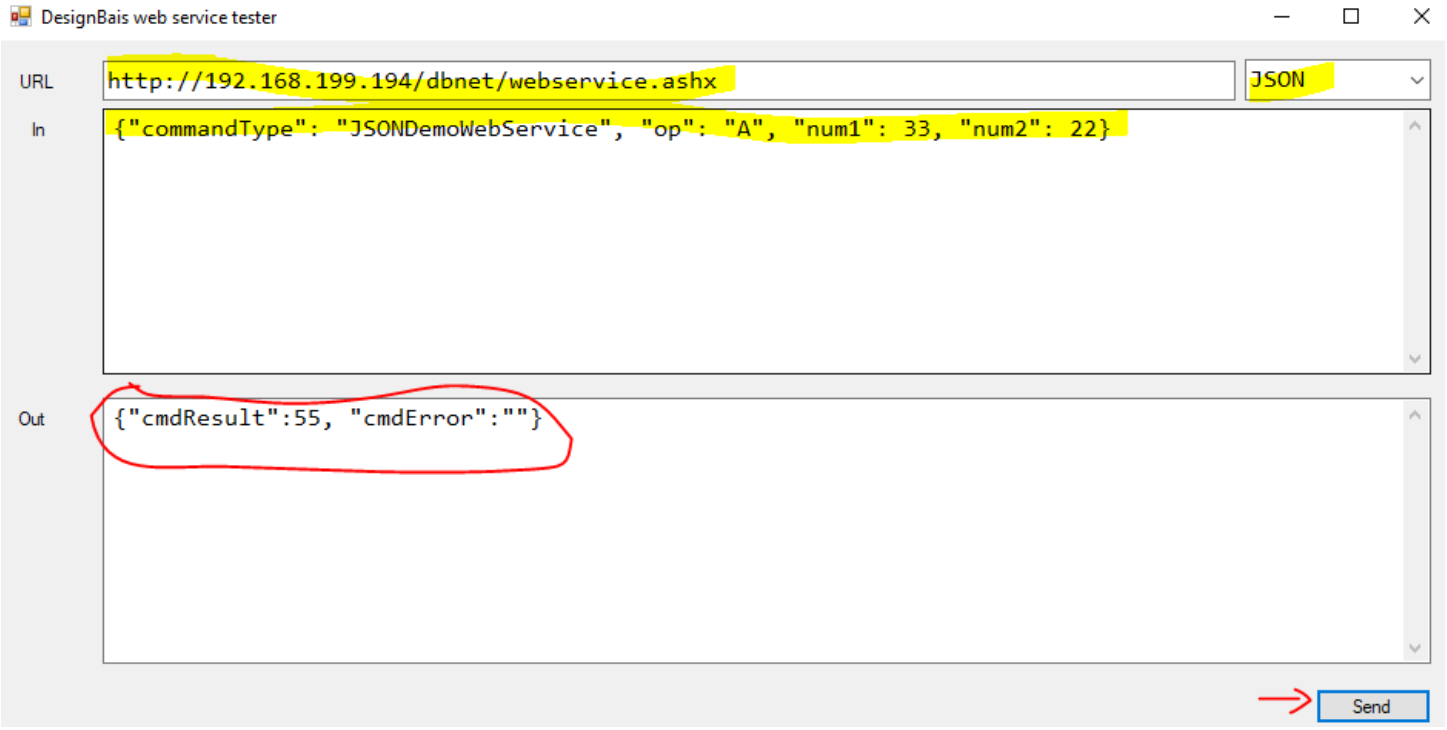
The web server (IIS) is responsible for authenticating the web service calls.
The caller can pass credentials in clear text but in that case SSL is recommended.

Developers can set up a web service that accepts JSON if the correct http content-type (application/json) is used. You can send and receive JSON without an XML envelope or JSON container. Use the "*application/json*" http header in your consumer (i.e. the caller.) Inbound JSON calls do not require the <dbWebService> tag but XML and SOAP calls do.

Picking JSON adds the header content-type.

```
031: <remoteService><header name="Content-Type"><![CDATA[application/json; chars
et=utf-8]]></header><wsUrl><![CDATA[http://192.168.199.194/dbnet/webservice.ashx
]]></wsUrl><wsData><![CDATA[{"commandType": "JSONDemoWebService", "op": "A", "nu
m1": 33, "num2": 22}]]></wsData></remoteService></screenData>
```

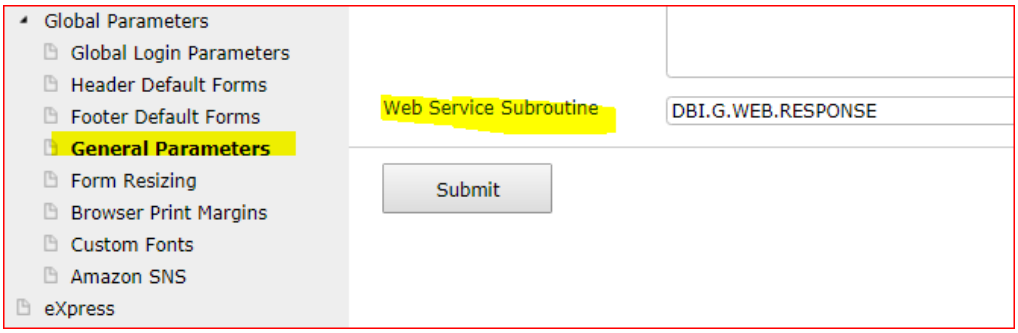
Example of JSON web service result:



Web Service Subroutine

All web service requests run through a sub-routine. The name of the subroutine must be specified in the *Web Service Subroutine* field in the *Global Parameters* maintenance form as shown below.
There is a template for this subroutine called *DBI.G.WEB.RESPONSE* in the DBINET file. This template should be copied to an application library, renamed, and amended as required by developers. The new routine name must be entered into the *Web Service Subroutine* field in Global General Parameters.

Note that when upgrading to a later release of DesignBais the standard version of this template (DBI.G.WEB.RESPONSE) will be installed and recataloged. If the standard version has been modified and utilised then your custom modifications will be lost.



SUBROUTINE DBI.G.WEB.RESPONSE(DATAIN.XML,DATAOUT.XML)

The incoming XML string (<x>new test</x>) is in DATAIN.XML.
The outbound string (<myData>abc</myData>) is in DATAOUT.XML.

The query string will be in the COMMON variable DBW3CQSTRING.

An error message using IERR.TEXT = “this is the error message” is passed via <error><![CDATA[this is the error message]]></error>.

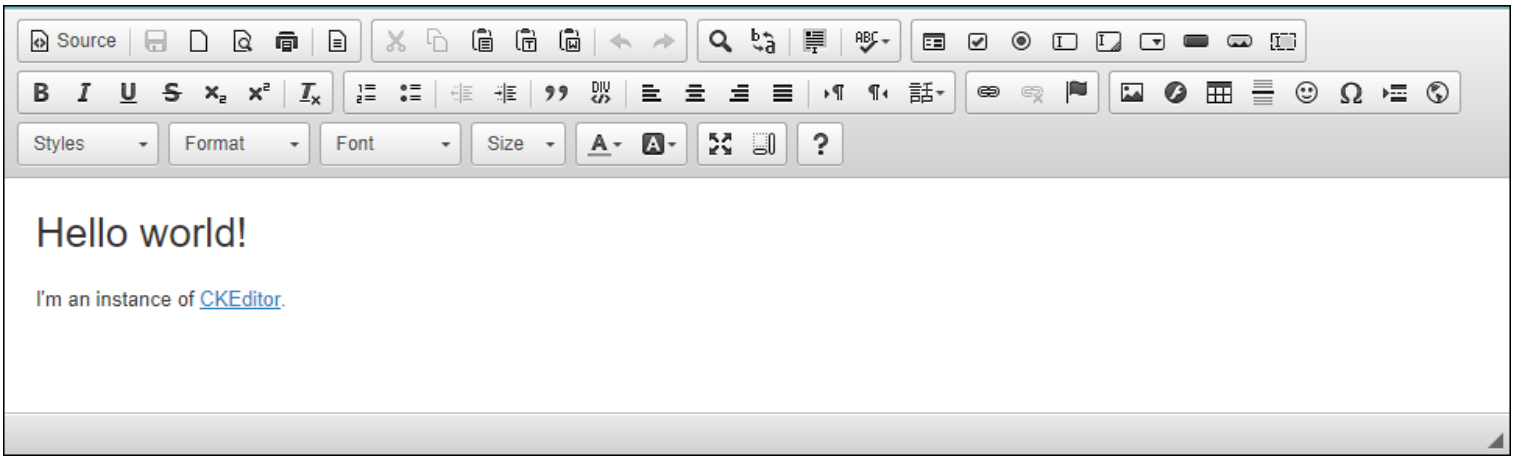
Chapter 19 – HTML Editor

HTML/RTF Editor

The RTFEDIT function was originally designed to invoke an RTF Editor. The main issue with this function was that the RTF editor needed to be installed on the end-users computer. A HTML editor is a simpler option as it can be invoked natively on all browsers, without any installation.

The RTFEDIT function has been modified to invoke a HTML editor. The editor is the CKEditor and full details are located here:

https://ckeditor.com/docs/ckeditor4/latest/guide/dev_installation.html



Usage:

```
DBRTFEDIT = "HTML"
DBRTFEDIT<2> = Header for the HTML editor.
DBRTFEDIT<3> = HTML string to pass
DBRTFEDIT<4> = Read Only flag ('true' or 'false') (null is equivalent to false)
DBRTFEDIT<5> = Include Print Button ('true' or 'false')   (null is equivalent to false)

DBRTFRESPONSE<1,1> = Program to call when editor is exited
DBRTFRESPONSE<1,2> = Eventsource supplied.
```

Example:

```
DBRTFEDIT = "HTML"
DBRTFEDIT<2> = "Client Maintenance Notes"
DBRTFEDIT<3> = "Hello editor. I can be plain text, or I can be HTML.
DBRTFEDIT<4> = 'false'
DBRTFEDIT<5> = 'true'

DBRTFRESPONSE = "DB.I.DBCLIENT":VM:"B.HTML"
```

When editor is closed, the program nominated in DBRTFRESPONSE<1,1> will be invoked. The eventsource will be as nominated in DBRTFRESPONSE<1,2>

```
PROCESS.EVENT = "HTMLEDIT"
PROCESS.EVENTSOURCE = DBRTFRESPONSE<1,2>
```

The HTML string returned from the editor will be stored in DBVALUE.
If DBVALUE is null this indicates that no change was made to the string passed to the editor.

The following example will assist in setting up a form to call the HTML Editor.

Refer to the list of fields below:

Field List

Field Property	Field Name	Field Read Use	Text	Col	Row	Col Span	Row Span	Field Section	Justification	Tab Index
TEXT	DBDESIGNERTEXT		Edit Text Files for DBIWEBSITE	0	0	100%	30	Main	[Default]	0
BUTTON	B.SEL.ID		DBWEBSITE Id	20	40	90	20	Main	[Default]	
INPUT	WEB.ID	DBOTHER.RECORD(1)	DBIWEBSITE Id	130	40	150	18	Main	[Default]	0
BUTTON	B.HTMLEDIT		Edit	450	40	90	20	Main	[Default]	0
BUTTON	B.SUBMIT		Submit	640	40	90	20	Buttons	[Default]	0
TEXT	TEXTONLY		Text of Record Being Edited	20	70	178	18	Main	[Default]	0
OUTPUT	WEB.EDIT.TEXT.WK	DBWORK	Text of Record Being Edited	20	90	800	18	Main	[Default]	0
INPUT	WEB.EDIT.TEXT	DBRECORD	Text of Record Being Edited	20	150	800	450	Hidden	[Default]	0

In this example the text to be edited is read into DBOTHER.RECORD(1).

DBRECORD is used to pass the text to and from the HTML Editor.

The work field WEB.EDIT.TEXT.WK is the output field used to display the text to be edited. It contains an HTML string. Ensure that "Encode HTML" is set to "No" on this field on the form.
The form presents like this:

Clicking the 'Edit' button opens the HTML Editor:

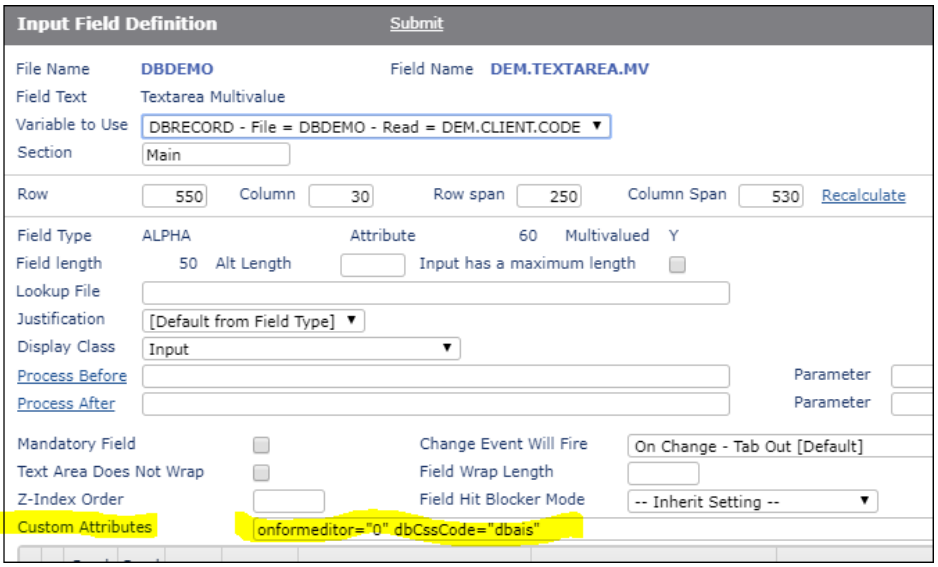
The basic code to handle the processing is shown below:

```
AFTER.READ:
      BEGIN CASE
         CASE SCREEN.NO = "WEBEDIT"
            TEXT = DBOTHER.RECORD(1)
            TEXT = CHANGE(TEXT,AM,'<br />')
            DBRECORD<WEB.EDIT.TEXT> = TEXT
            DBENABLEFIELD<1> = SCREENROOT
            DBENABLEFIELD<2> = 'B.SUBMIT'
            DBENABLEFIELD<3> = 'D'
            GOSUB DISPLAY.HTML
      END CASE
      RETURN

BUTTON:
      BEGIN CASE
         CASE EVENTSOURCE = "B.HTMLEDIT" AND SCREEN.NO = "WEBEDIT"
            DBVALUE = ""
            READONLY = 'false'
            PRINTBUTTON = 'true'
            TEXT = DBRECORD<WEB.EDIT.TEXT>
            DBRTFEDIT    = "HTML"
            DBRTFEDIT<2> = "Text from ":DBKEY
            DBRTFEDIT<3> = TEXT
            DBRTFEDIT<4> = READONLY
            DBRTFEDIT<5> = PRINTBUTTON
            DBRTFRESPONSE = "DB.I.WEB"
            DBRTFRESPONSE<1,2> = "B.HTMLEDIT"
      END CASE
      RETURN

BEFORE.WRITE:
      BEGIN CASE
         CASE EVENTSOURCE = "B.SUBMIT" AND SCREEN.NO = "WEBEDIT"
            TEXT = DBRECORD<WEB.EDIT.TEXT>
            TEXT = CHANGE(TEXT,'<br />',AM)
            POS = INDEX(TEXT,"<body>",1) + 6
            ENDPOS = INDEX(TEXT,"</body>",1)
            TEXTLEN = ENDPOS - POS
            TEXT = TEXT[POS,TEXTLEN]
            IF NOT(TEXT = "") THEN DBOTHER.RECORD(1) = TEXT
            DBWORK<WEB.EDIT.TEXT.WK> = ""
      END CASE
      RETURN

HTMLEDIT:
      BEGIN CASE
         CASE SCREEN.NO = "WEBEDIT" AND THIS.PARAMETER = "B.HTMLEDIT"
            IF DBVALUE="" THEN
               DBDS<-1> = 'Text not changed.'
               DBENABLEFIELD<1> = SCREENROOT
               DBENABLEFIELD<2> = 'B.SUBMIT'
               DBENABLEFIELD<3> = 'D'
               RETURN
            END
            IF DBVALUE # DBORIGINAL.RECORD<WEB.EDIT.TEXT> THEN
               DBENABLEFIELD<1> = SCREENROOT
               DBENABLEFIELD<2> = 'B.SUBMIT'
               DBENABLEFIELD<3> = 'E'
            END
            * Store the HTML Page string
            IF DBVALUE # DBRECORD<WEB.EDIT.TEXT> THEN
```

```
            DBRECORD<WEB.EDIT.TEXT> = DBVALUE
            GOSUB DISPLAY.HTML
         END

      END CASE
      RETURN

DISPLAY.HTML:
      * To keep a valid HTML string on a form we need to use an iFrame to
      * tell the web server to ignore the content held in the field being
      * updated.
      *
      * Thus we will always have a pair of fields, one to hold the HTML content
      * and one to hold the iframe that references the content field.
      *
      * Create the frame in an Output Only field. This field may be hidden
      * if the HTML page is not to be displayed.
      *
      * The id "myiframe" needs to be unique for each pairing.
      *
      * The iframe must have the content field name in the dbSource attribute
      * in order to link the two fields.
      *
      * The iframe can be styled using a class name or inline styles as below.
      *
      SRC.FLD.NAME = "DBIWEBSITE WEB.EDIT.TEXT"      ;* filename and field name
      SRC.FLD.NAME = CHANGE(SRC.FLD.NAME,"."," ")    ;* replace "." and "-" with a space
      SRC.FLD.NAME = CHANGE(SRC.FLD.NAME,"-"," ")
      SRC.FLD.NAME = ICONV(SRC.FLD.NAME,"MCT")       ;* text/sentence case
      SRC.FLD.NAME = CHANGE(SRC.FLD.NAME," ","")     ;* no spaces in final name "DbclientDbcNote"
      HTML.STR = '<iframe id="myiframe" style="width:530px;height:400px;border:2px solid #cccccc;border-
radius:7px;" dbSource="':SRC.FLD.NAME:'" />'
      DBWORK<WEB.EDIT.TEXT.WK> = HTML.STR   ;* can be any attribute
      * Convert "  " to "  " which is what is returned by editor for multiple spaces.
      LOOP UNTIL INDEX(DBRECORD<WEB.EDIT.TEXT>,"  ",1) = 0 DO
         DBRECORD<WEB.EDIT.TEXT> = CHANGE(DBRECORD<WEB.EDIT.TEXT>,"  ","  ")
      REPEAT
      RETURN
```

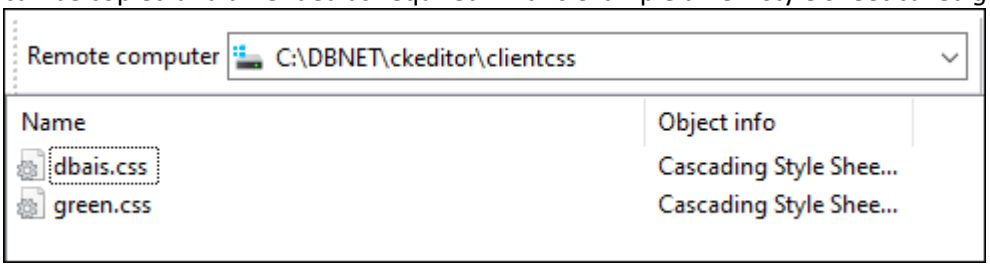
On Form HTML Editor

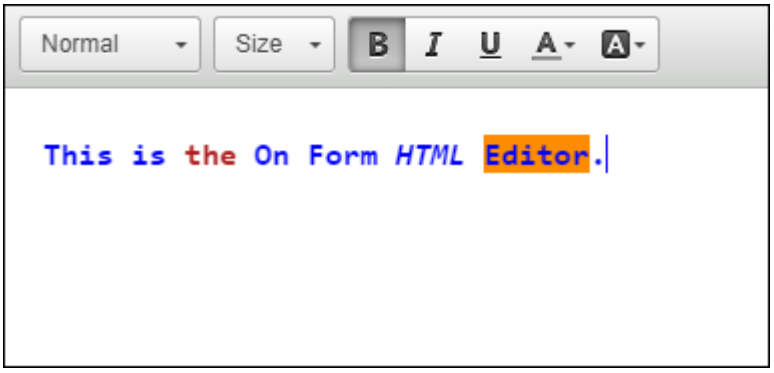
The On Form HTML Editor can be invoked by adding a custom attribute *onformeditor="0"* to an input field on a form. To set the style sheet add the attribute *dbCssCode="dbais"* where *"dbais"* defines the required style sheet.



The row span of the field must be at least 120 px. The col span must be at least 350 px. The field must be set as a text area field (essentially it must have the multivalue flag checked - see *text area* in this manual). On form editor fields currently cannot be hidden, disabled or collapsed.

The editor is based on the CKEditor. There is a folder in the DesignBais website called *ckeditor*. The *dbais.css* style sheet can be copied and amended as required. In this example a new style sheet called *green.css* has been created.





Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Chapter 20 – Development Checklist

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Development Checklist

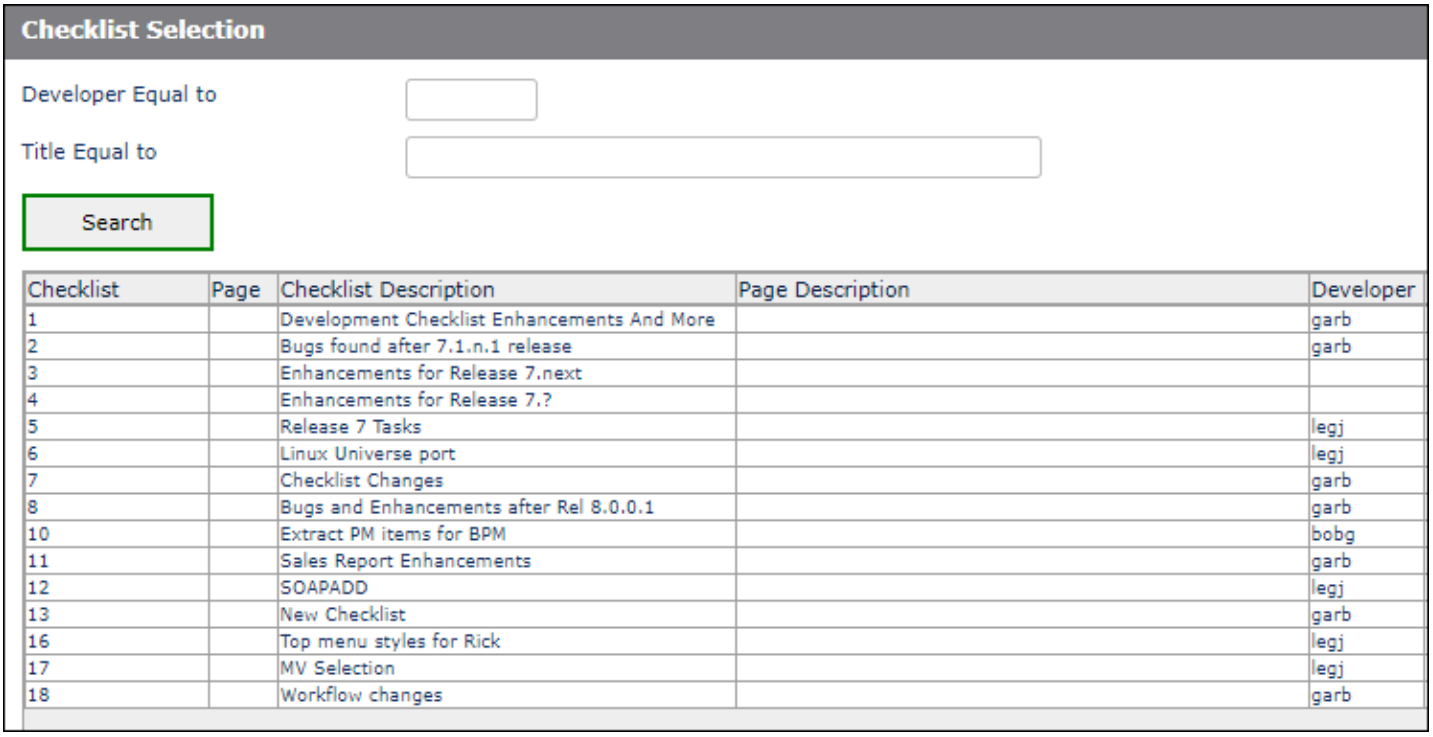
This function provides an easy method to deploy DesignBais forms and reports.

A Checklist is a list of Ids of forms, fields, reports and any other DesignBais entity that make up a software application.

A Checklist can contain multiple pages and each page can hold multiple DesignBais forms and reports. It is our recommendation, however, that a new page is used for each form or report. This facilitates the management of the deployment of changes.

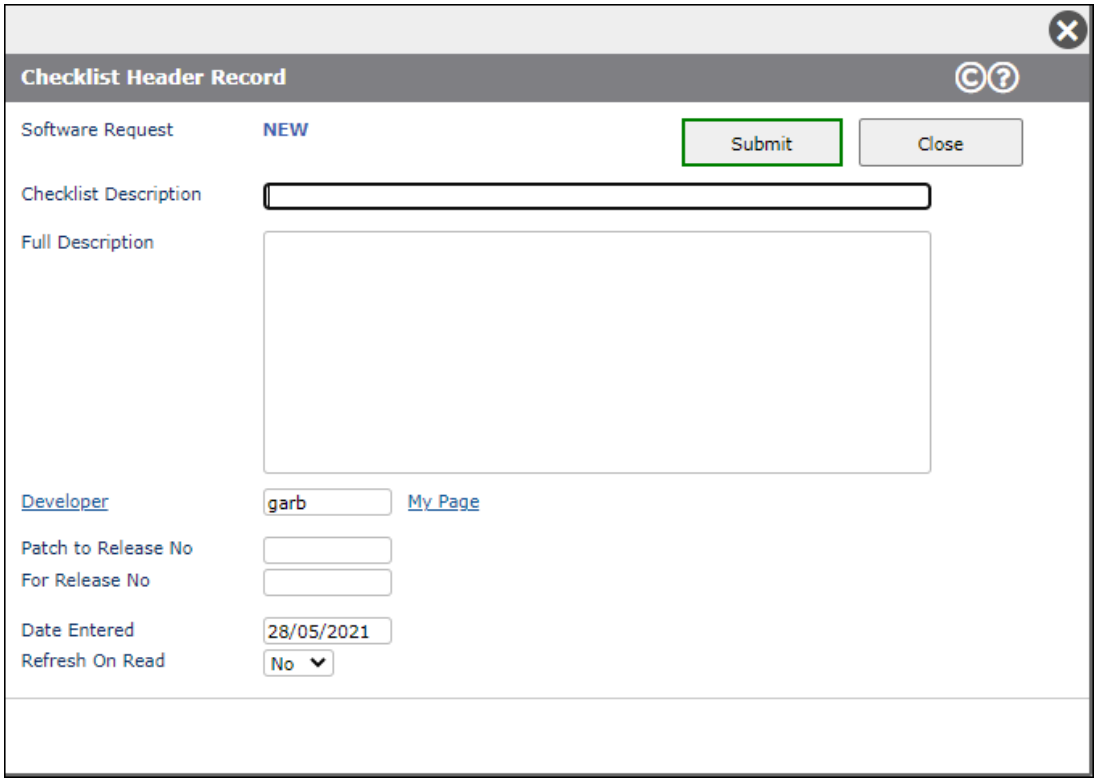
Selecting the *Development Checklist* option from the DesignBais Tools side menu displays the following form:

Checklist — Clicking this button runs a selection process to display existing checklists. Use the *Developer* or *Title Equal To* fields to refine your search results.



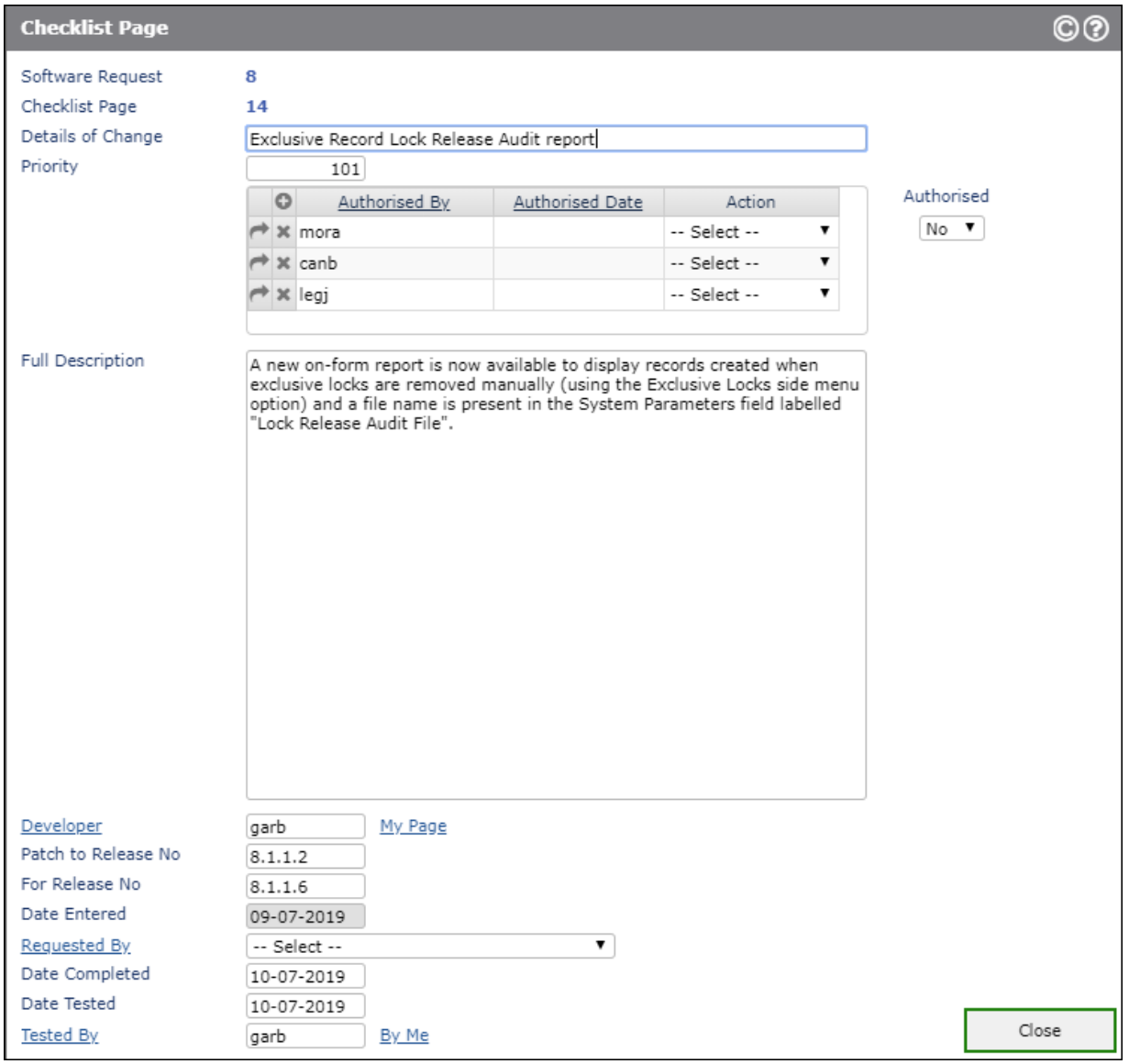
Checklist	Page	Checklist Description	Page Description	Developer
1		Development Checklist Enhancements And More		garb
2		Bugs found after 7.1.n.1 release		garb
3		Enhancements for Release 7.next		
4		Enhancements for Release 7.?		
5		Release 7 Tasks		legj
6		Linux Universe port		legj
7		Checklist Changes		garb
8		Bugs and Enhancements after Rel 8.0.0.1		garb
10		Extract PM items for BPM		bobg
11		Sales Report Enhancements		garb
12		SOAPADD		legj
13		New Checklist		garb
16		Top menu styles for Rick		legj
17		MV Selection		legj
18		Workflow changes		garb

New [button] — Clicking this button opens the Checklist Header Form. Here you can enter details pertaining to the Software Request. A checklist may have multiple pages. Typically a checklist would reflect all the changes for a patch release. The next available checklist number is allocated after clicking the Submit button on this form.



Checklist Page — Clicking this button runs a selection process to display existing checklist pages for the selected checklist. Each page of a checklist may contain all of the field/form/report records associated with a DesignBais form or a DesignBais Report.

New [button] — Clicking this button opens the Checklist Page form. Here you can enter details pertaining to the checklist page.



The details of the checklist page are displayed.

Refresh this Checklist On Read

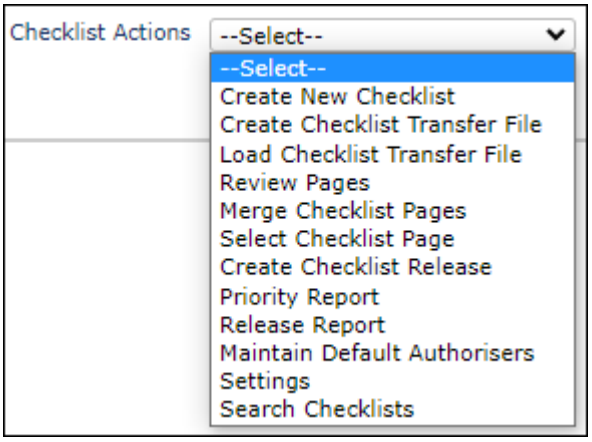
Check this box if you want the checklist page to be refreshed on read. This means that all forms and other processes recorded on the checklist will be reviewed and any new elements that have been added to them will be added to the list of items on the checklist.

Note that top and side menu items referenced by a form are not added to the checklist. This is because menus can contain security information, either initially or by this refresh.

Add Subroutines to Checklist

Check this box if you want the subroutine names from forms and other processes to be added to the checklist.

Checklist Actions — Displays a dropdown selection list of actions associated with a checklist header.



Create New Checklist — Creates a new checklist header. The same action can be initiated by clicking the *New* button.

Create Checklist Transfer File — See detailed description below.

Load Checklist Transfer File — See detailed description below.

Review Pages

Merge Checklist Pages

Select Checklist Page

Create Checklist Release

Priority Report

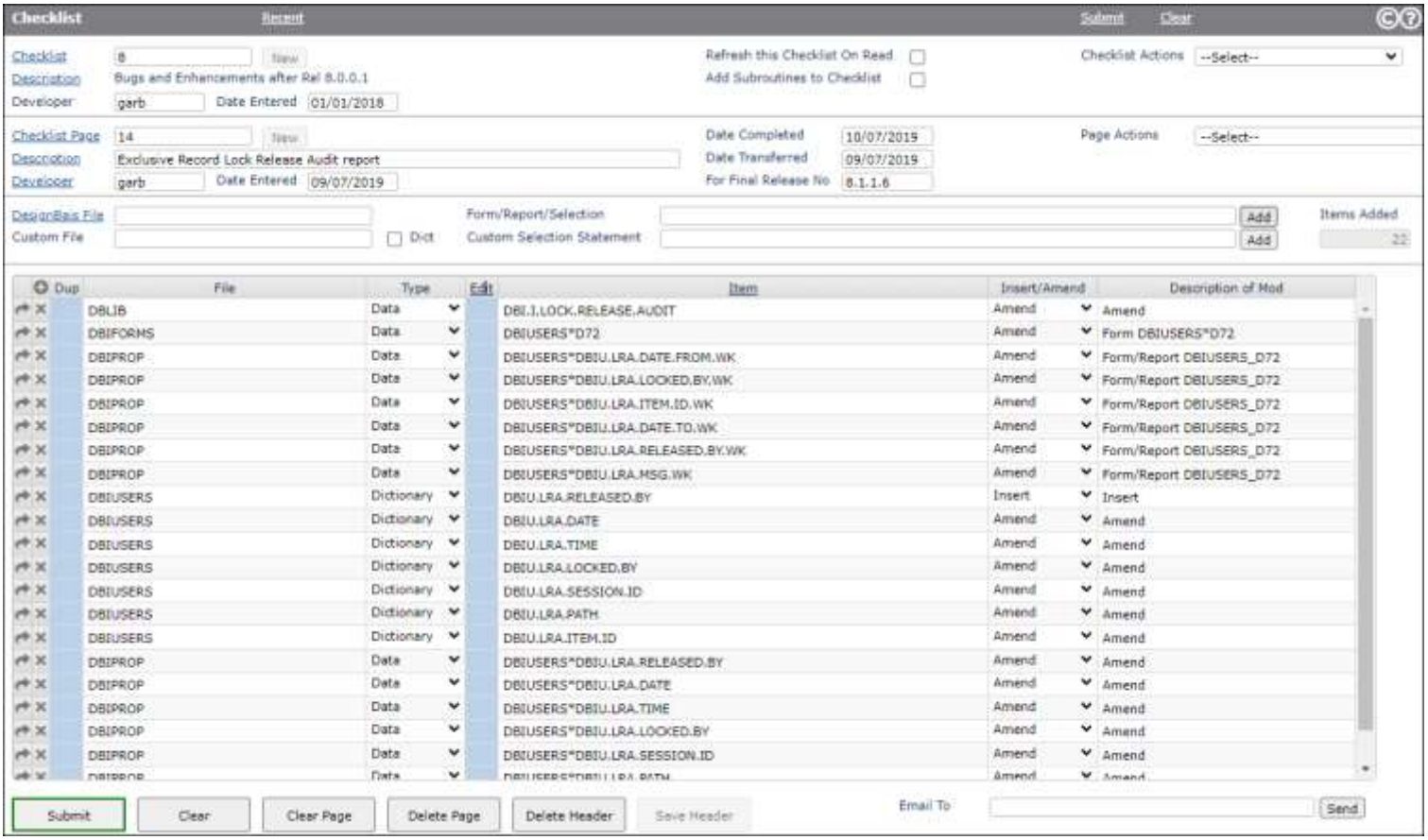
Release Report

Maintain Default Authorisers

Settings

Search Checklists

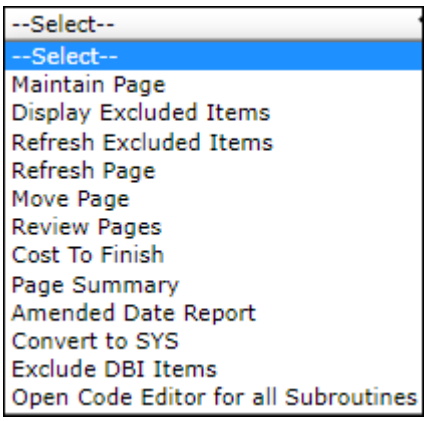
Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3



Prompts

Checklist	The number assigned to the checklist. A checklist may have multiple pages. Typically a checklist would reflect all the changes for a patch release.
Description	The description of the checklist header record. Click the hyperlink to open a form to maintain the header details.
Developer	The user id of the user that created the checklist header.
Date Entered	The date the checklist header was created.
Checklist Page	Click the hyperlink to select an existing checklist page. Otherwiase click the *New* button to create a new page.
Description	The description of the checklist page record. Each page of a checklist may contain all of the field/form/report records associated with a DesignBais form or a DesignBais Report. Click the hyperlink to open a form to maintain the static page details.
Developer	The user id of the user that created the checklist page. Click the hyperlink to select a user id from the DBIUSERS File.
Date Entered	The date the checklist page was created.
Priority	The priority can be assigned. Used for sorting tasks.
Developer	The user code of the developer that created the page.
Date Completed	The date the task was completed.
Tested By	The user code of the developer that tested the change.
Date Tested	The date the task was tested.
Date Transferred	The date the task was transferred to a test or production environment.

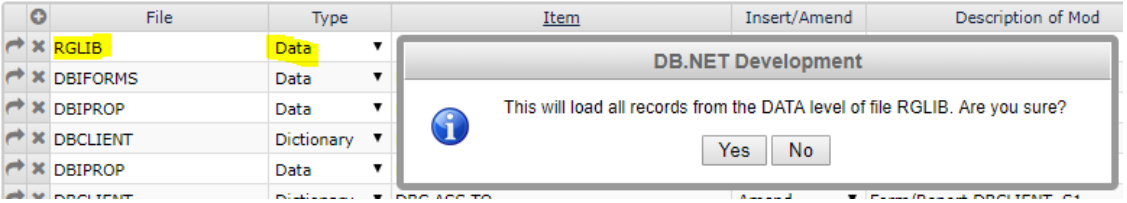
Page Actions	The dropdown selection list provides access to functions related to a checklist page.
Maintain Page	Opens a form to allow you to maintain the static page details.
Display Excluded Items	Displays a list of excluded items on thei checklist page.
Refresh Excluded Items	Refreshes the excluded status of items on the page.
Refresh Page	
Move Page	
Review Pages	See below.
Cost To Finish	
Page Summary	
Amended Data Report	
Convert to SYS	This option is for DesignBais use.
Exclude DBI Items	This option is for DesignBais use.
Open all Subroutines in Code Editor	On the user record for developers there is a *Basic Library File* list. All rows on the current checklist page that have a file name from this list are considered to be subroutines. All such records will be opened in the Code Editor. The same function is available by clicking the *Edit* button in column 4 of the grid header row.
DesignBais File	The name of the DesignBais file for the form, report or selection to be added to the checklist.
Form/Report/Selection	The name of the form or report or selection process.
Add to List	Click this button to add all of the records associated with the Form, Report or Selection to the checklist page. **This does not include program references. Developers need to add these manually.**
Items Added	This field displays the number of items added to the checklist page as a result of clicking the Add to List button. On initial display of the checklist page it shows the total number of items currently in the page.
Dup	Click to duplicate the file name from the row above.



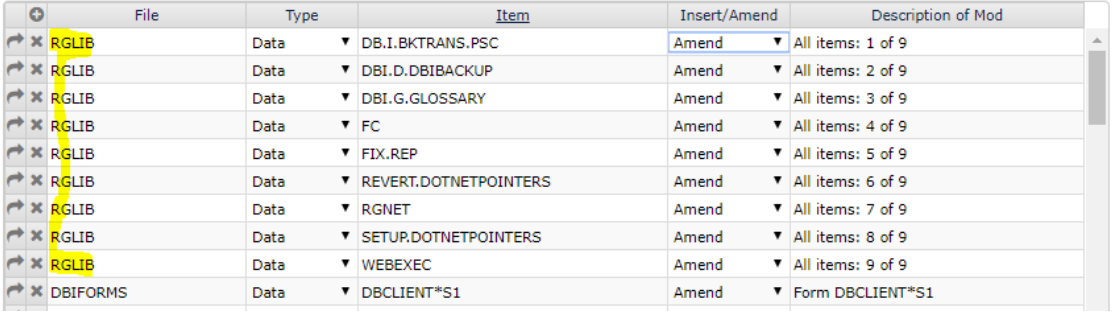
In order to add a form to a checklist page enter the name of the file that the form relates to in the DesignBais File field, then enter the name of the form in the Form/Report/Selection field. For example to add the form DBCLIENT*DEMO to a checklist enter DBCLIENT in the DesignBais File field and DEMO in the Form/Report/Selection field. Then click “Add to List”

To add all the field properties and dictionaries for a file enter the name of the file in the *DesignBais File* field and leave the *Form/Report/Selection* field blank. Then click “*Add to List*”.

To add all items from a file that is not a defined DesignBais file (that is it is not a record on DBIFILES) enter the file name in the grid *File* column, select either *Dict* or *Data*, then enter an asterisk in the *Item* column. You will be prompted:



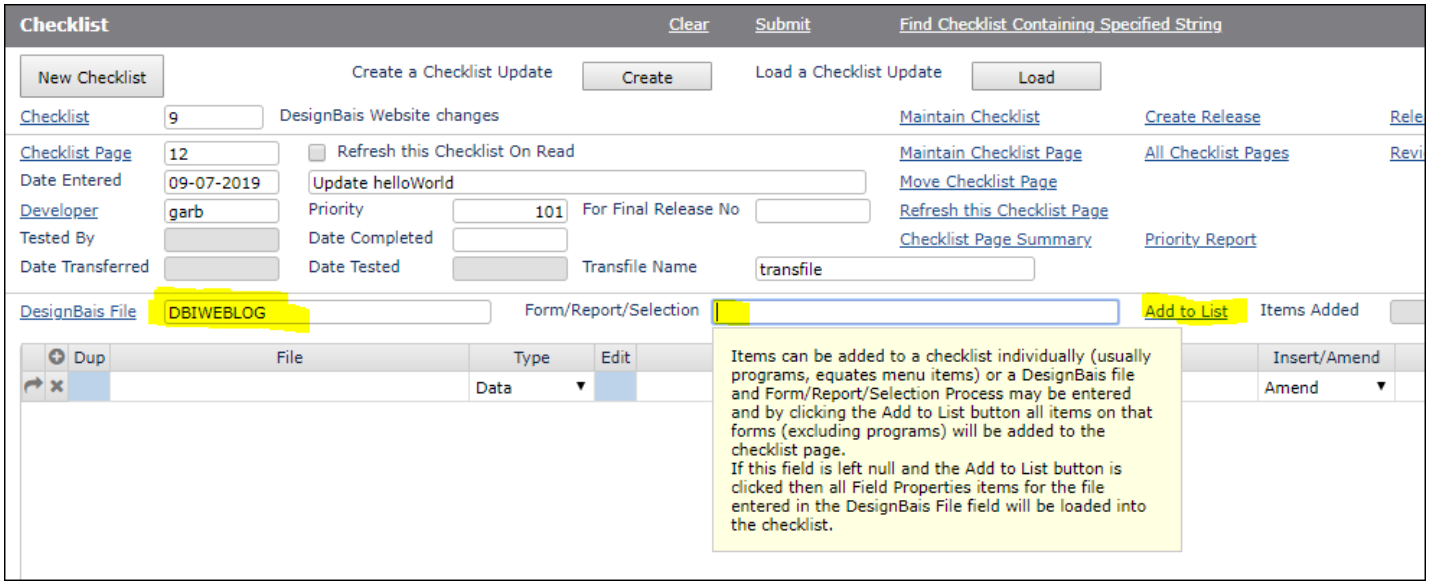
Clicking *Yes* will load all items into the checklist.



To enter the name of the basic program associated with a form you must use the grid. Enter the program file name in the *File* column of the grid, for example DBLIB. Select the type “Data”. Enter the name of the program in the Item column.

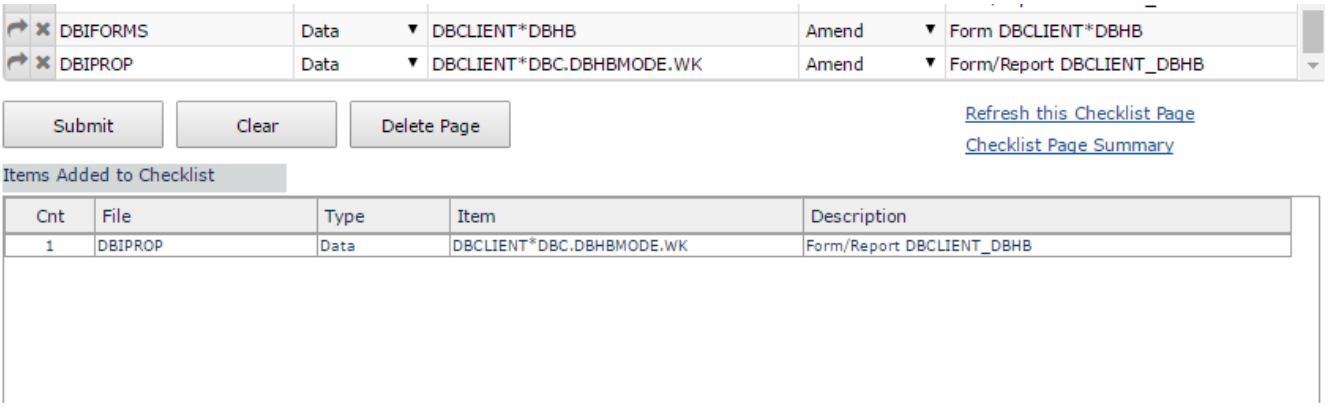
Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Another method of adding all items is shown below:



Refresh Page

Clicking this button will check all items on the checklist page (Files, Forms, Reports and Selections) and ensure that all items associated with each form and report are included in the checklist. Missing or new items are added. This allows the developer to pick up new fields added to forms for example without having to re-enter the form names again.



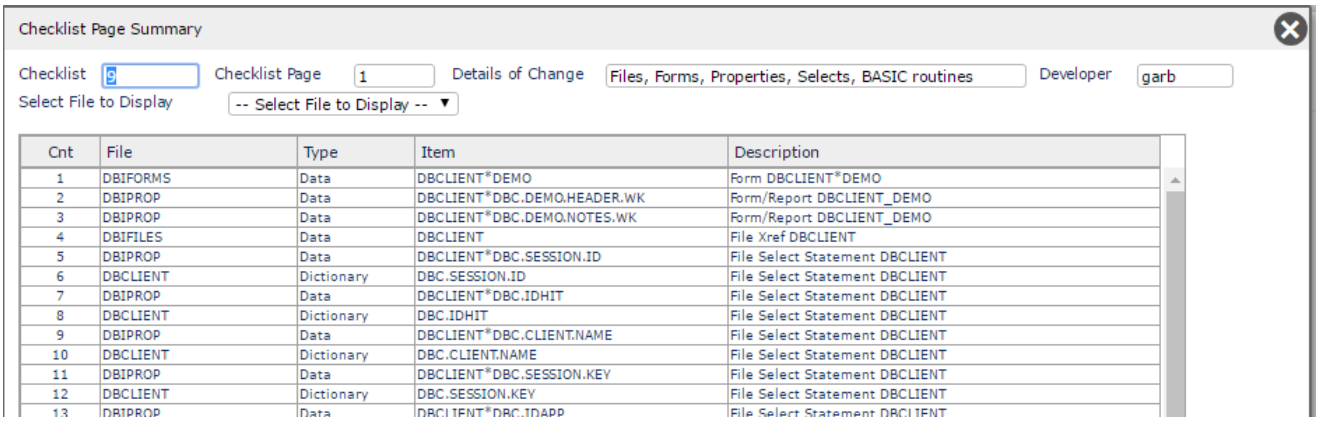
A report of the the items added is displayed as shown above.

Move Page

This option allows you to move checklist pages to another (existing) checklist.

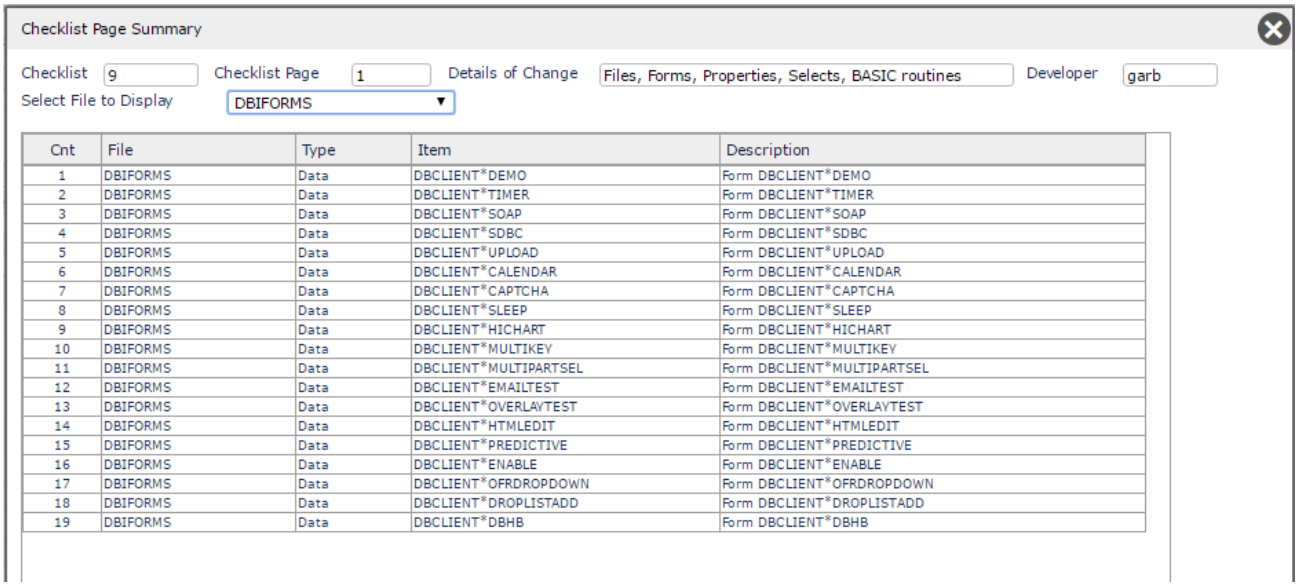
Checklist Page Summary

This button displays the entire contents of the checklist page in an On-form Report. The header column is clickable to allow sorting of each column.



Cnt	File	Type	Item	Description
1	DBIFORMS	Data	DBCLIENT*DEMO	Form DBCLIENT*DEMO
2	DBIPROP	Data	DBCLIENT*DBC.DEMO.HEADER.WK	Form/Report DBCLIENT_DEMO
3	DBIPROP	Data	DBCLIENT*DBC.DEMO.NOTES.WK	Form/Report DBCLIENT_DEMO
4	DBIFILES	Data	DBCLIENT	File Xref DBCLIENT
5	DBIPROP	Data	DBCLIENT*DBC.SESSION.ID	File Select Statement DBCLIENT
6	DBCLIENT	Dictionary	DBC.SESSION.ID	File Select Statement DBCLIENT
7	DBIPROP	Data	DBCLIENT*DBC.IDHIT	File Select Statement DBCLIENT
8	DBCLIENT	Dictionary	DBC.IDHIT	File Select Statement DBCLIENT
9	DBIPROP	Data	DBCLIENT*DBC.CLIENT.NAME	File Select Statement DBCLIENT
10	DBCLIENT	Dictionary	DBC.CLIENT.NAME	File Select Statement DBCLIENT
11	DBIPROP	Data	DBCLIENT*DBC.SESSION.KEY	File Select Statement DBCLIENT
12	DBCLIENT	Dictionary	DBC.SESSION.KEY	File Select Statement DBCLIENT
13	DBIPROP	Data	DBCLIENT*DBC.IDAPP	File Select Statement DBCLIENT

Then using the “Select File to Display” dropdown the developer can display items relating to each file. In the example below the report shows just the forms that are on the checklist page.



Cnt	File	Type	Item	Description
1	DBIFORMS	Data	DBCLIENT*DEMO	Form DBCLIENT*DEMO
2	DBIFORMS	Data	DBCLIENT*TIMER	Form DBCLIENT*TIMER
3	DBIFORMS	Data	DBCLIENT*SOAP	Form DBCLIENT*SOAP
4	DBIFORMS	Data	DBCLIENT*SDBC	Form DBCLIENT*SDBC
5	DBIFORMS	Data	DBCLIENT*UPLOAD	Form DBCLIENT*UPLOAD
6	DBIFORMS	Data	DBCLIENT*CALENDAR	Form DBCLIENT*CALENDAR
7	DBIFORMS	Data	DBCLIENT*CAPTCHA	Form DBCLIENT*CAPTCHA
8	DBIFORMS	Data	DBCLIENT*SLEEP	Form DBCLIENT*SLEEP
9	DBIFORMS	Data	DBCLIENT*HICHART	Form DBCLIENT*HICHART
10	DBIFORMS	Data	DBCLIENT*MULTIKEY	Form DBCLIENT*MULTIKEY
11	DBIFORMS	Data	DBCLIENT*MULTIPARTSEL	Form DBCLIENT*MULTIPARTSEL
12	DBIFORMS	Data	DBCLIENT*EMAILTEST	Form DBCLIENT*EMAILTEST
13	DBIFORMS	Data	DBCLIENT*OVERLAYTEST	Form DBCLIENT*OVERLAYTEST
14	DBIFORMS	Data	DBCLIENT*HTMLEDIT	Form DBCLIENT*HTMLEDIT
15	DBIFORMS	Data	DBCLIENT*PREDICTIVE	Form DBCLIENT*PREDICTIVE
16	DBIFORMS	Data	DBCLIENT*ENABLE	Form DBCLIENT*ENABLE
17	DBIFORMS	Data	DBCLIENT*OFRDROPDOWN	Form DBCLIENT*OFRDROPDOWN
18	DBIFORMS	Data	DBCLIENT*DROPLISTADD	Form DBCLIENT*DROPLISTADD
19	DBIFORMS	Data	DBCLIENT*DBHB	Form DBCLIENT*DBHB

Review Pages

This button opens form to allow you to view all or selected pages for selected software requests.

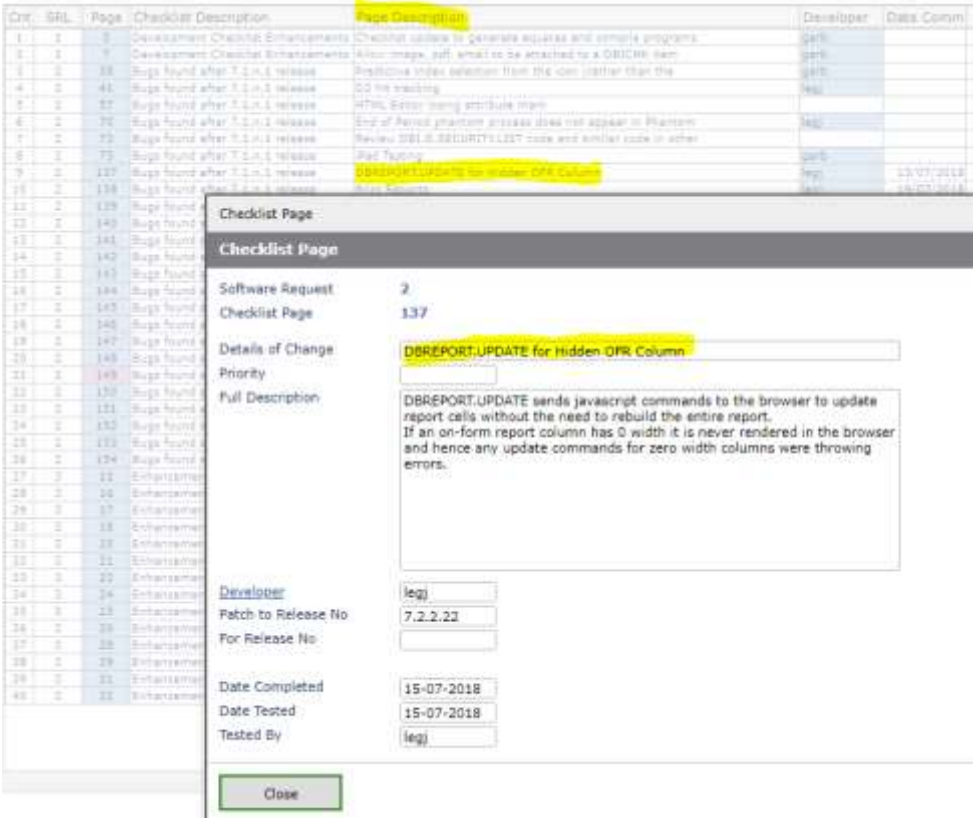
Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

To review pages for a single software request enter the checklist id in the ***Checklist*** field at the top of the form. If this field is null then all checklists will be selected.

The display can be filtered by selecting from the *Display Filter Options* dropdown. The *Exclude Transferred Pages* option allows you to display only those tasks that are yet to be completed.

Clicking a selected row on the *Page Description* column opens the checklist page maintenance form to allow you to review or update the details.

Clicking on the *Final Release* column places the value in the clicked row into the *Final Release* filter field and refreshes the report to display only those rows that match this value. Click again to remove the filter.



Clicking the *Close* button on this form will update any changes made to the Checklist Page details. The background color on rows that are reviewed is changed to indicate which rows have been accessed.

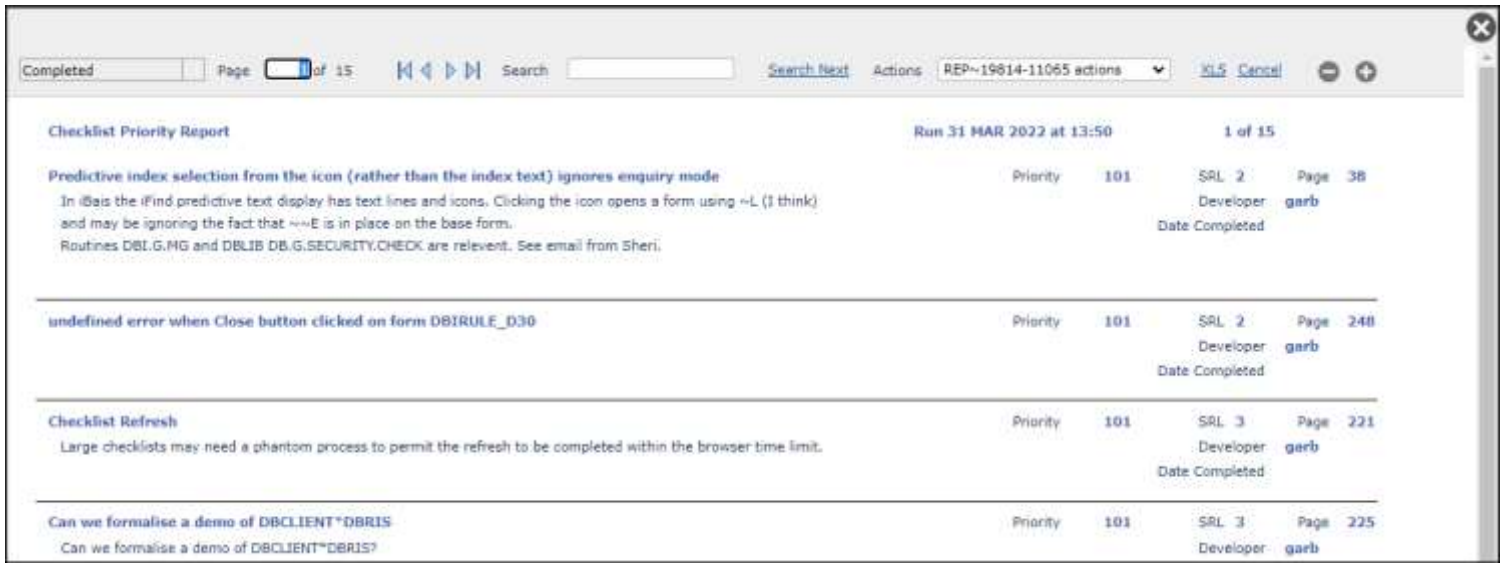
Clicking on the *Developer* column filters the display to show tasks for the selected developer.

Clicking the *Full Description* check box then clicking the *Refresh* button displays the full page description in the On-form Report.

Enter a value in the *Final Release* field then click the *Refresh* button to display pages for that release.

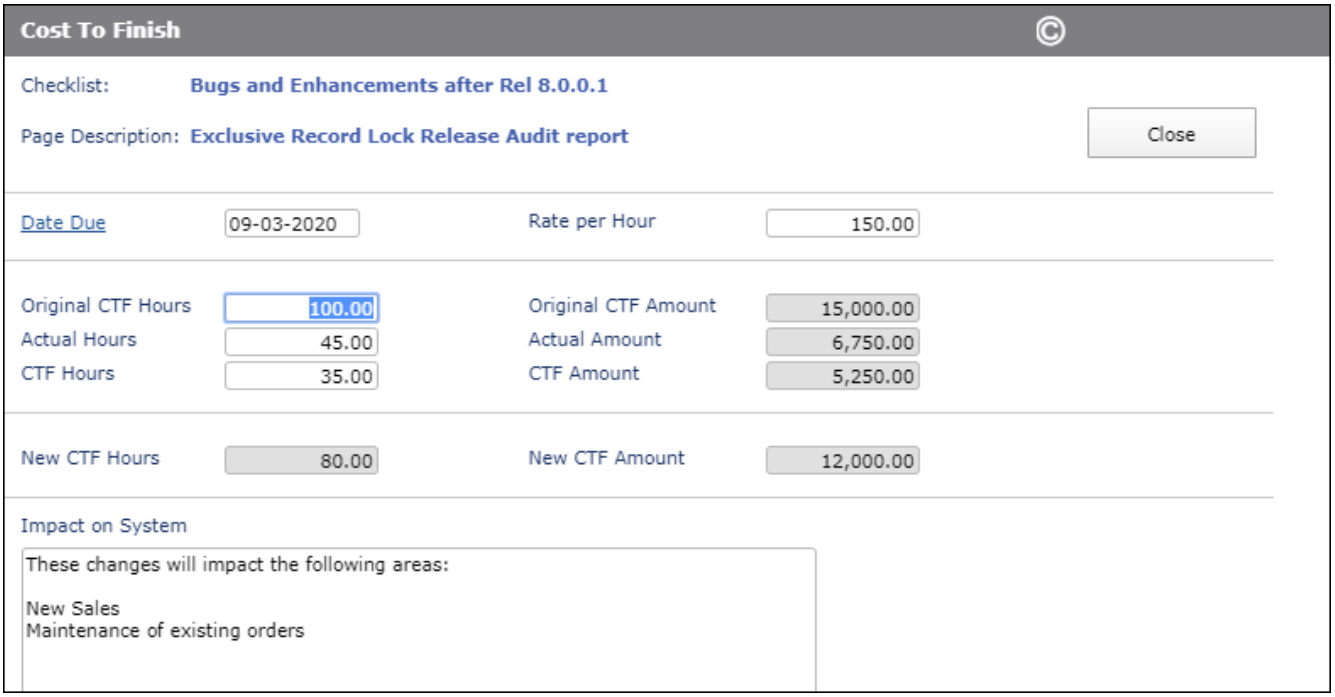
Priority Report

This button runs a report for selected *Developer* and selected *Priority*.



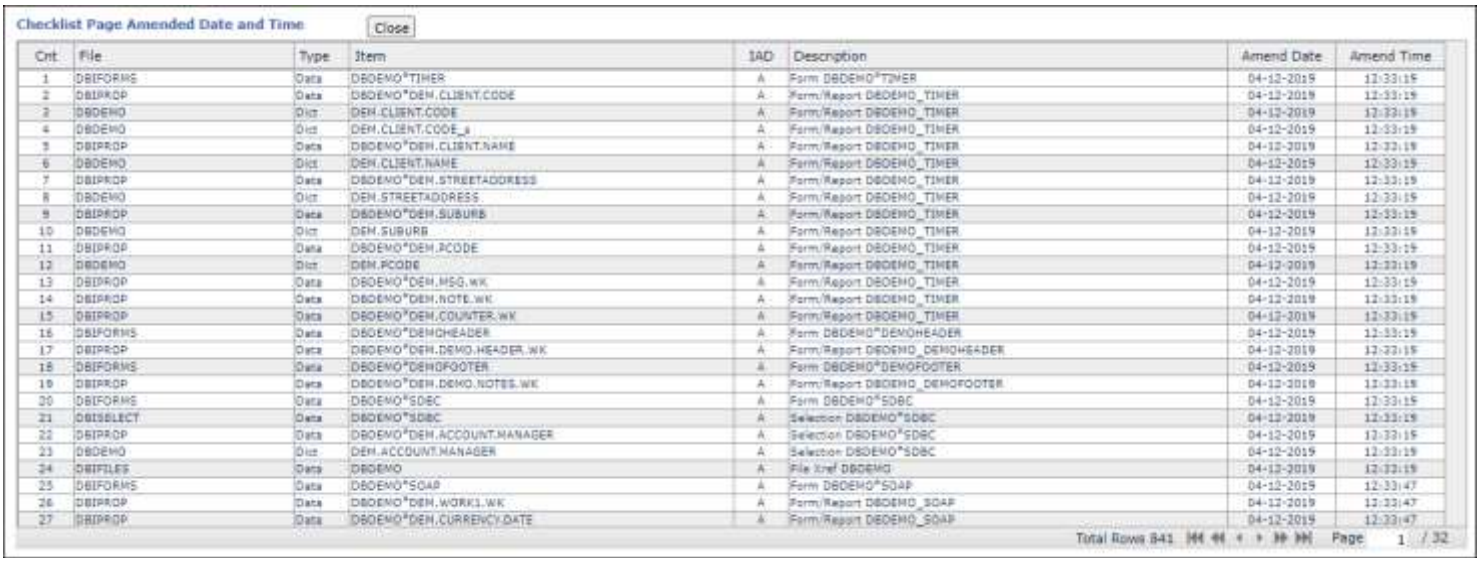
Cost to Finish

This screen displays fields that can be used to track project cost to finish. It is very basic. The following snip shows that after 45 hours of work on a task that is estimated to require 100 hours, the new estimate of the cost to finish is 35 hours. This implies a new total cost of 80 hours which is less than the 100 hours original estimate.



Amended Date Report

Creates an on-form report of the items in the currently displayed checklist page sorted by *Amend Date*.



Cnt	File	Type	Item	IAD	Description	Amend Date	Amend Time
1	DBIFORMS	Data	DBDEMO*TIMER	A	Form DBDEMO*TIMER	04-12-2019	12:33:19
2	DBIPROP	Data	DBDEMO*DEM.CLIENT.CODE	A	Form/Report DBDEMO_TIMER	04-12-2019	12:33:19
3	DBDEMO	Dict	DEM.CLIENT.CODE	A	Form/Report DBDEMO_TIMER	04-12-2019	12:33:19
4	DBDEMO	Dict	DEM.CLIENT.CODE_a	A	Form/Report DBDEMO_TIMER	04-12-2019	12:33:19
5	DBIPROP	Data	DBDEMO*DEM.CLIENT.NAME	A	Form/Report DBDEMO_TIMER	04-12-2019	12:33:19
6	DBDEMO	Dict	DEM.CLIENT.NAME	A	Form/Report DBDEMO_TIMER	04-12-2019	12:33:19
7	DBIPROP	Data	DBDEMO*DEM.STREETADDRESS	A	Form/Report DBDEMO_TIMER	04-12-2019	12:33:19
8	DBDEMO	Dict	DEM.STREETADDRESS	A	Form/Report DBDEMO_TIMER	04-12-2019	12:33:19
9	DBIPROP	Data	DBDEMO*DEM.SUBURB	A	Form/Report DBDEMO_TIMER	04-12-2019	12:33:19
10	DBDEMO	Dict	DEM.SUBURB	A	Form/Report DBDEMO_TIMER	04-12-2019	12:33:19
11	DBIPROP	Data	DBDEMO*DEM.PCODE	A	Form/Report DBDEMO_TIMER	04-12-2019	12:33:19
12	DBDEMO	Dict	DEM.PCODE	A	Form/Report DBDEMO_TIMER	04-12-2019	12:33:19
13	DBIPROP	Data	DBDEMO*DEM.MSG.WK	A	Form/Report DBDEMO_TIMER	04-12-2019	12:33:19
14	DBIPROP	Data	DBDEMO*DEM.NOTE.WK	A	Form/Report DBDEMO_TIMER	04-12-2019	12:33:19
15	DBIPROP	Data	DBDEMO*DEM.COUNTER.WK	A	Form/Report DBDEMO_TIMER	04-12-2019	12:33:19
16	DBIFORMS	Data	DBDEMO*DEMOHEADER	A	Form DBDEMO*DEMOHEADER	04-12-2019	12:33:19
17	DBIPROP	Data	DBDEMO*DEM.DEMO.HEADER.WK	A	Form/Report DBDEMO_DEMOHEADER	04-12-2019	12:33:19
18	DBIFORMS	Data	DBDEMO*DEMOFOOTER	A	Form DBDEMO*DEMOFOOTER	04-12-2019	12:33:19
19	DBIPROP	Data	DBDEMO*DEM.DEMO.NOTES.WK	A	Form/Report DBDEMO_DEMOFOOTER	04-12-2019	12:33:19
20	DBIFORMS	Data	DBDEMO*SDBC	A	Form DBDEMO*SDBC	04-12-2019	12:33:19
21	DBISELECT	Data	DBDEMO*SDBC	A	Selection DBDEMO*SDBC	04-12-2019	12:33:19
22	DBIPROP	Data	DBDEMO*DEM.ACCOUNT.MANAGER	A	Selection DBDEMO*SDBC	04-12-2019	12:33:19
23	DBDEMO	Dict	DEM.ACCOUNT.MANAGER	A	Selection DBDEMO*SDBC	04-12-2019	12:33:19
24	DBIFILES	Data	DBDEMO	A	File Xref DBDEMO	04-12-2019	12:33:19
25	DBIFORMS	Data	DBDEMO*SOAP	A	Form DBDEMO*SOAP	04-12-2019	12:33:47
26	DBIPROP	Data	DBDEMO*DEM.WORK1.WK	A	Form/Report DBDEMO_SOAP	04-12-2019	12:33:47
27	DBIPROP	Data	DBDEMO*DEM.CURRENCY.DATE	A	Form/Report DBDEMO_SOAP	04-12-2019	12:33:47

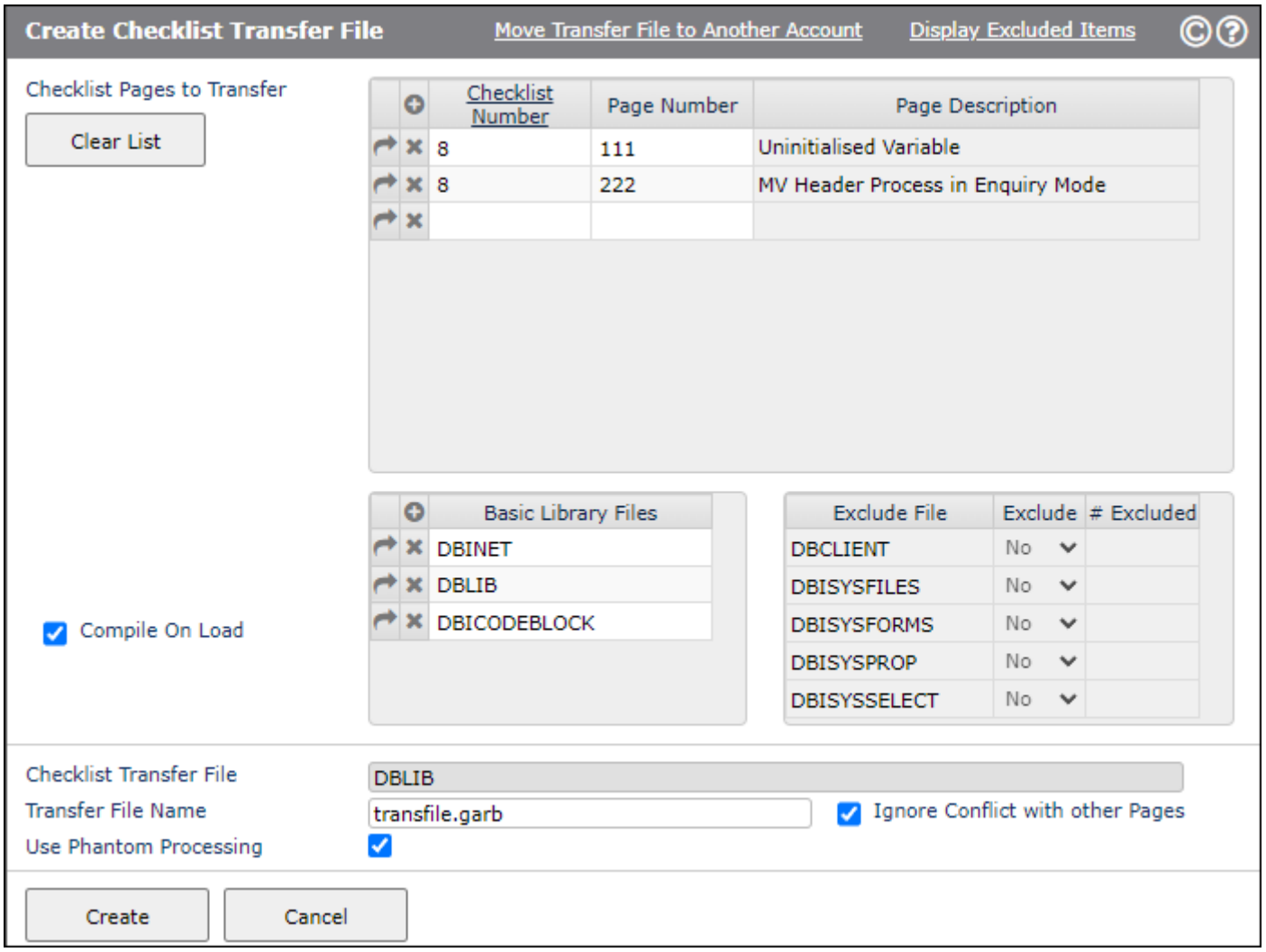
Click on column in the header row to sort by that column.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Create Checklist Transfer File

This option will load the following form.

This form is used to pack all of the required checklists and their respective pages into a single record. This single record can then be copied to a destination system and unpacked.

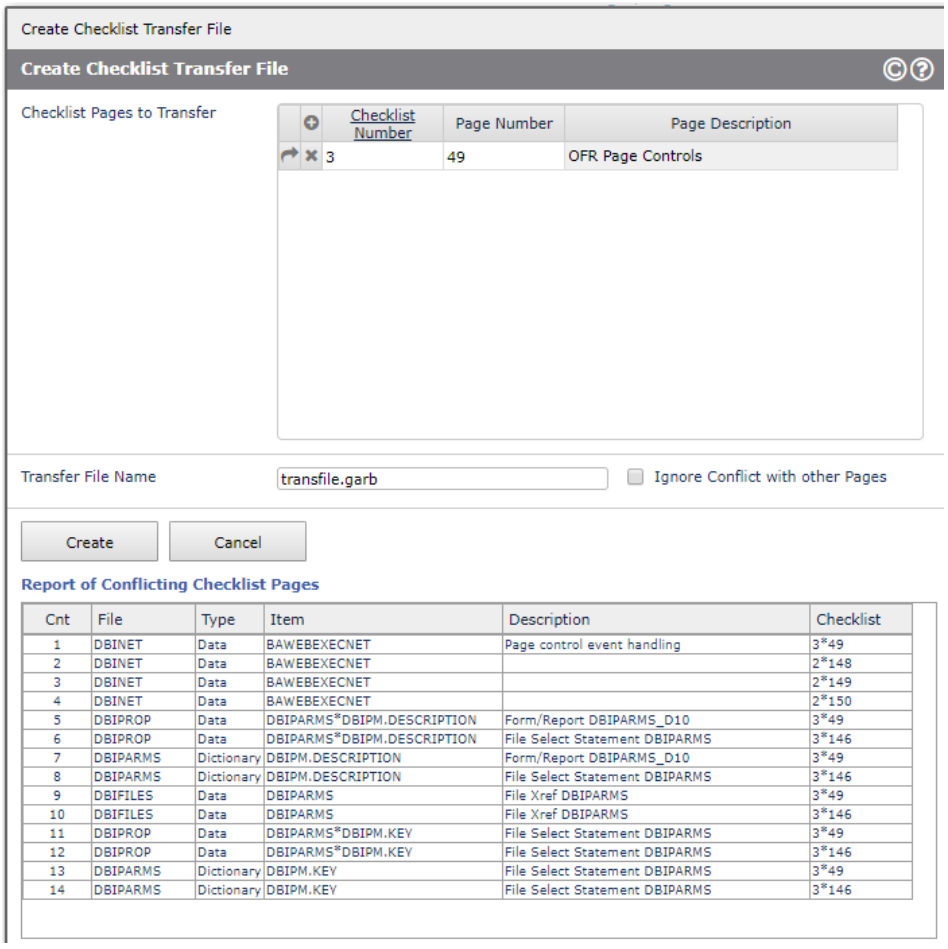


Checklist Number	Enter the checklist number
Page Number	Enter the Page Number(s) to be included into the pack.
Transfer File Name	Enter the name of the pack record that will be created. This will be created in the file DBLIB.
Basic Library Files	The default list is derived from your user record Default Basic Library files. Remove or add file names as required for the particular checklist update. Only applies to users with access to the Code Editor.
Compile On Load	Click this check box if you want loaded items, from any of the files in the Basic Library Files list, to be compiled after they are loaded in the target environment. Only applies to users with access to the Code Editor.
Ignore Conflict with	Click this check box if pages are to be loaded regardless of conflicts with other pages.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Other Pages

The default behaviour is to display a *Report of Conflicting Checklist Pages*. A conflicting page is defined as one which is not included in the list of pages to form the transfer file but which contains a software item that is present on a page that is included in this transfer file.

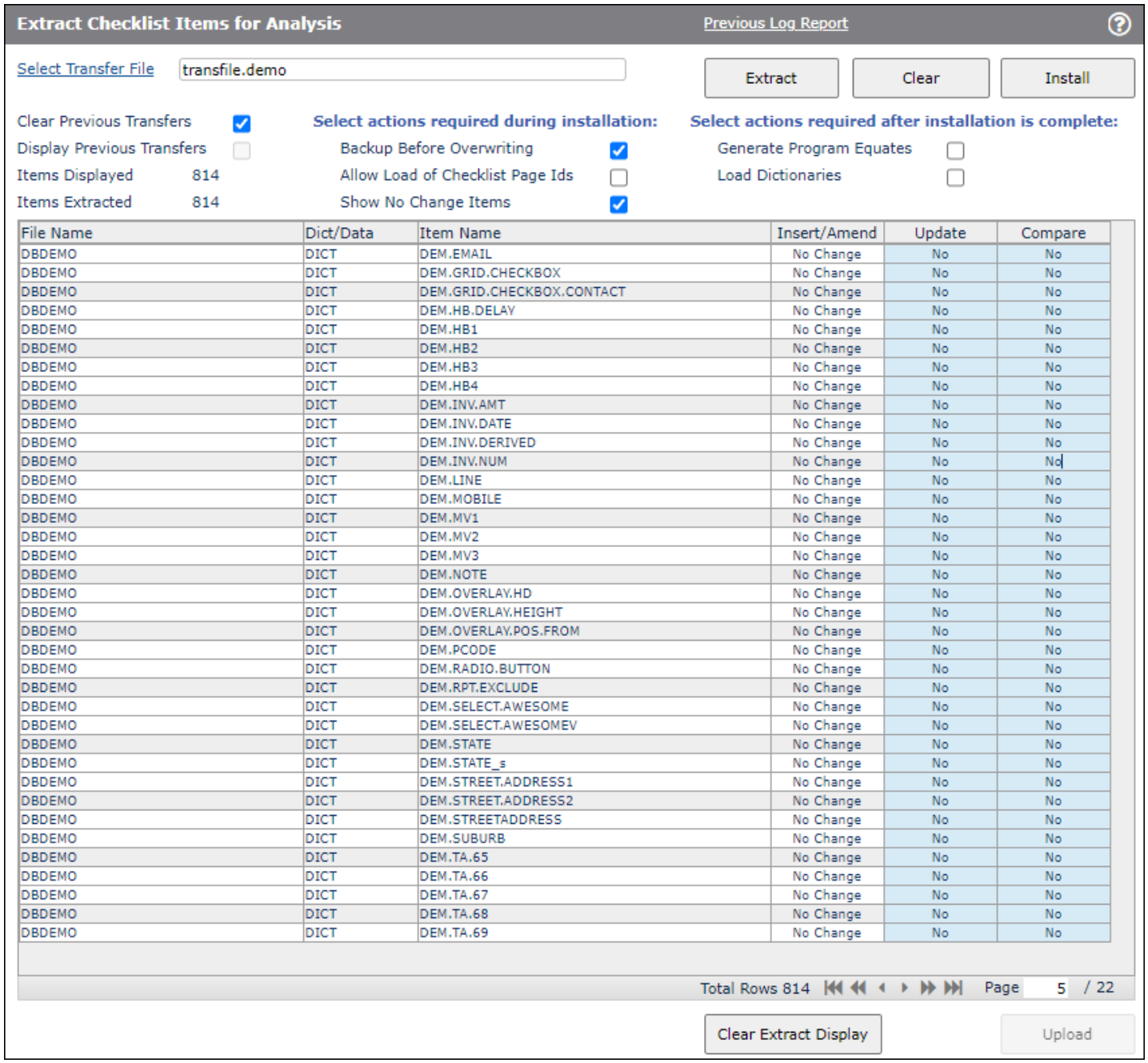


Cnt	File	Type	Item	Description	Checklist
1	DBINET	Data	BAWEBEXECNET	Page control event handling	3*49
2	DBINET	Data	BAWEBEXECNET		2*148
3	DBINET	Data	BAWEBEXECNET		2*149
4	DBINET	Data	BAWEBEXECNET		2*150
5	DBIPROP	Data	DBIPARMS*DBIPM.DESCRIPTION	Form/Report DBIPARMS_D10	3*49
6	DBIPROP	Data	DBIPARMS*DBIPM.DESCRIPTION	File Select Statement DBIPARMS	3*146
7	DBIPARMS	Dictionary	DBIPM.DESCRIPTION	Form/Report DBIPARMS_D10	3*49
8	DBIPARMS	Dictionary	DBIPM.DESCRIPTION	File Select Statement DBIPARMS	3*146
9	DBIFILES	Data	DBIPARMS	File Xref DBIPARMS	3*49
10	DBIFILES	Data	DBIPARMS	File Xref DBIPARMS	3*146
11	DBIPROP	Data	DBIPARMS*DBIPM.KEY	File Select Statement DBIPARMS	3*49
12	DBIPROP	Data	DBIPARMS*DBIPM.KEY	File Select Statement DBIPARMS	3*146
13	DBIPARMS	Dictionary	DBIPM.KEY	File Select Statement DBIPARMS	3*49
14	DBIPARMS	Dictionary	DBIPM.KEY	File Select Statement DBIPARMS	3*146

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Load Checklist Transfer File

Select this option to display the Extract Checklist Items for Analysis form. After selecting your transfer file you can review the items that will be installed in the target environment.



This form displays the contents of the transfer file before it is loaded and will compare items on the transfer file with those on the target system. Those items that differ will be flagged for update. Any item that differs can be compared to the version on the target system in order to highlight what will be changed if the transfer file is loaded.

Select Transfer File — Click this link to display the list of transfer files that are held in DBLIB or DBITRANSFILE. Select the required file.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Clear Previous Transfers

Click this check box to clear all checklist extract items from the DBICHKEXTRACT file that belong to your user id. These items will have a key structure like *weblogon|seqno*. You can load multiple transfer files before installing a set of changes. If this is the objective then you must uncheck this flag for all transfer files following the first transfer file.

Backup Before Overwriting

Check this field to copy target account items to DBIBACKUP prior to loading changed items from a checklist transfer file. Only items that are to be overwritten with an amended version will be backed up.

Allow Load of Checklist Page Ids

This option allows the developer to load checklist items (DBICHK records) from the transfer file to the target DBICHK file.

Show No Change Items

Display items from the checklist that match the item in the target environment, as well as displaying new and amended items.

Generate Program Equates

If set then all program equates for all files with properties included in this checklist will be re-built after loading this checklist update.

Load Dictionaries

If set then dictionary items for all files with properties included in this checklist will be re-loaded after loading this checklist update.

Extract (Button)

Click this button to extract the items from the transfer file. No updating occurs. Items are displayed in the on-form report.

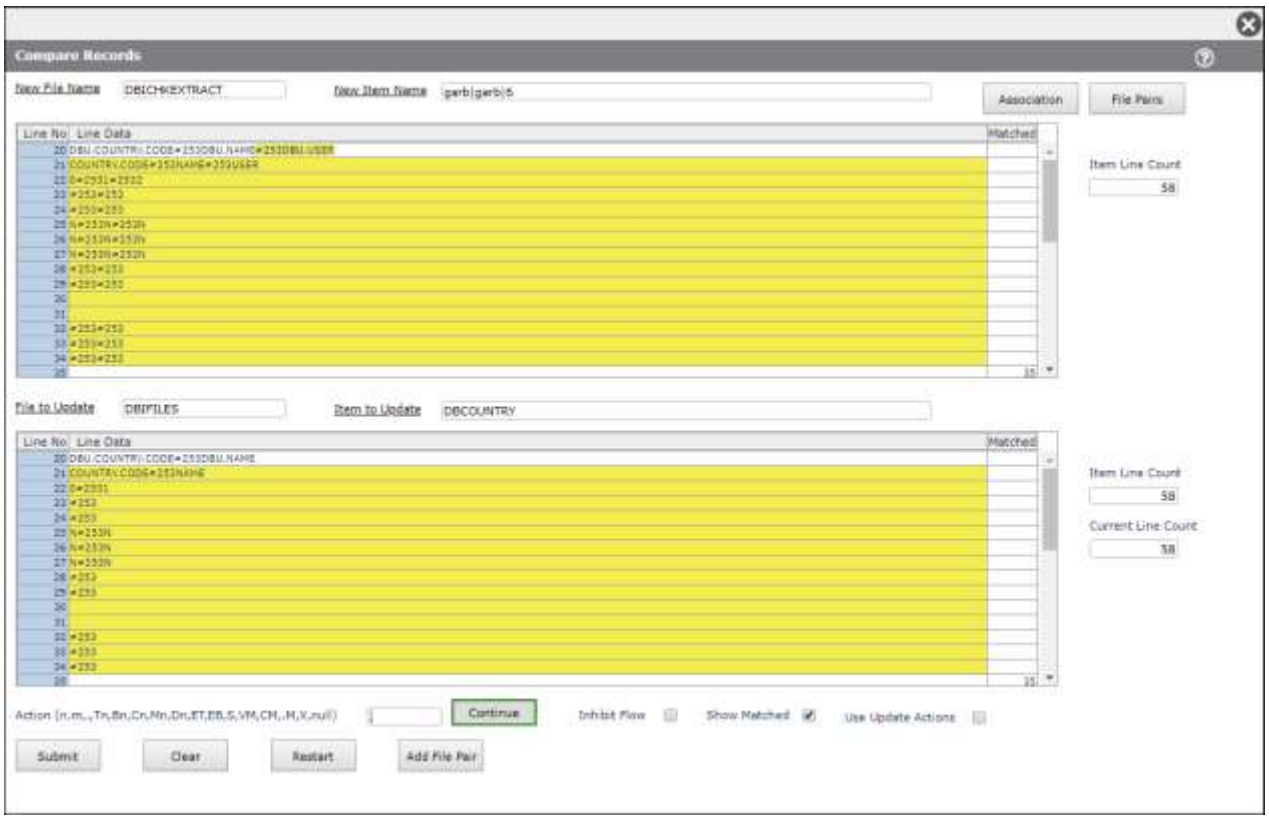
Transfer File Display

Update Column

If an item is flagged as *Amend* or *Insert* then ypu can choose whether you wish to update the target and the *Compare* flag is set to *Yes* to allow the developer, if required, to review what changes will be applied if the transfer file is installed.

Items are flagged *No Change* if the item on the transfer file matches the item in the target environment. These items can be hidden by unchecking the *Show No Change Items* check box. The *Show No Change Items* check box controls whether items on the transfer file that match the equivalent items on the target system are displayed. If they are displayed then by default the *Update* option and the *Compare* option for matching items are set to *No*.

The *Update* and *Compare* columns can be toggled. If the *Insert/Amend* column is set to *Amend* then clicking the row in the *Compare* column will display the compare form with the two items loaded. The display halts at the first difference between the two items.



Install (Button)

Click the *Install* button to load the items from the transfer file flagged for update into the target environment. A log of changes is displayed.

Routines will be compiled and cataloged if they are on a file that is in the *Basic Library File* list in the User Maintenance form for the logged in user, or if the filename ends with *LIB* (in upper case). For example compilation will be attempted for an item in a file called *TESTLIB*.

Extract Checklist Items for Analysis — Previous Log Report

Select Transfer File

Extract | Clear | Install

Clear Previous Transfers ☑
Display Previous Transfers ☐
Items Displayed 71
Items Extracted 71

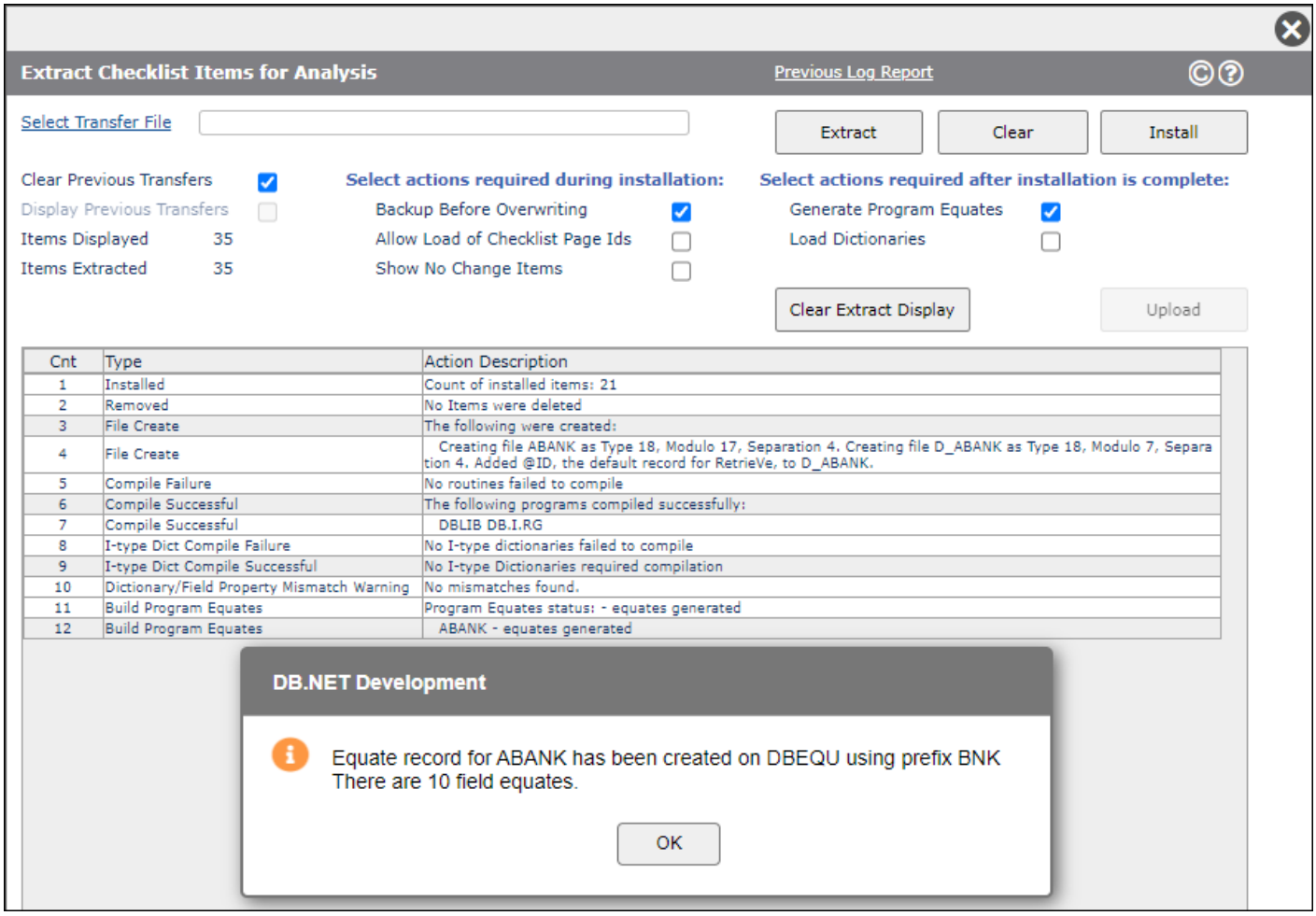
Select actions required during installation:
Backup Before Overwriting ☑
Allow Load of Checklist Page Ids ☐
Show No Change Items ☐

Select actions required after installation is complete:
Generate Program Equates ☐
Load Dictionaries ☐

Clear Extract Display | Upload

Cnt	Type	Action Description
1	Installed	Count of installed items: 9
2	Removed	Count of deleted items: 0
3	File Create	No Files were created
4	Compile Failure	No routines failed to compile
5	Compile Successful	The following programs compiled successfully:
6	Compile Successful	DBLIB DB.G.GET.USER
7	Compile Successful	DBLIB DB.I.DBISURVEY
8	Compile Successful	DBLIB DB.I.DBIWEBSITE
9	Compile Successful	COMPILE.DICT DBIREGISTRATIONS
10	Build Program Equates	Program Equates status: - the Generate Program Equates checkbox was not ticked.
11	Build Program Equates	DBIREGISTRATIONS - the Generate Program Equates checkbox was not ticked.
12	Build Program Equates	DBISURVEY - the Generate Program Equates checkbox was not ticked.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

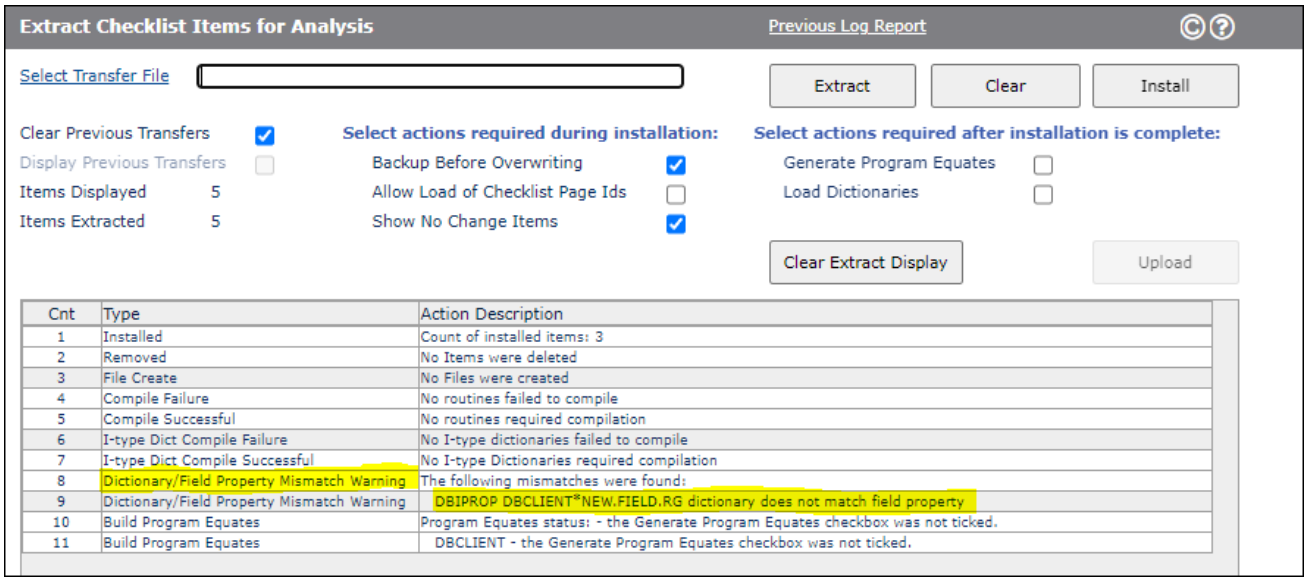


I-type dictionary items are recompiled after loading using the *COMPILE.DICT* command.

The “Generate Program Equates” and “Load Dictionaries” checkboxes will force a refresh of the DBIFILES record field property list in the target account by forcing a dummy amend of a field property in each file that is represented in the checklist. From release 8.8 onwards the dummy amend is automatic for all files that have field properties on the checklist regardless of the checkbox settings.

If a checklist includes a new file then it will be created automatically. The checklist row containing the DBIFILES record will be set to “Update” (Yes) even if the default setting is “No” since the DBIFILES record contains the *Equates From File* and *Equates Prefix* which are required in the target environment DBIFILES record.

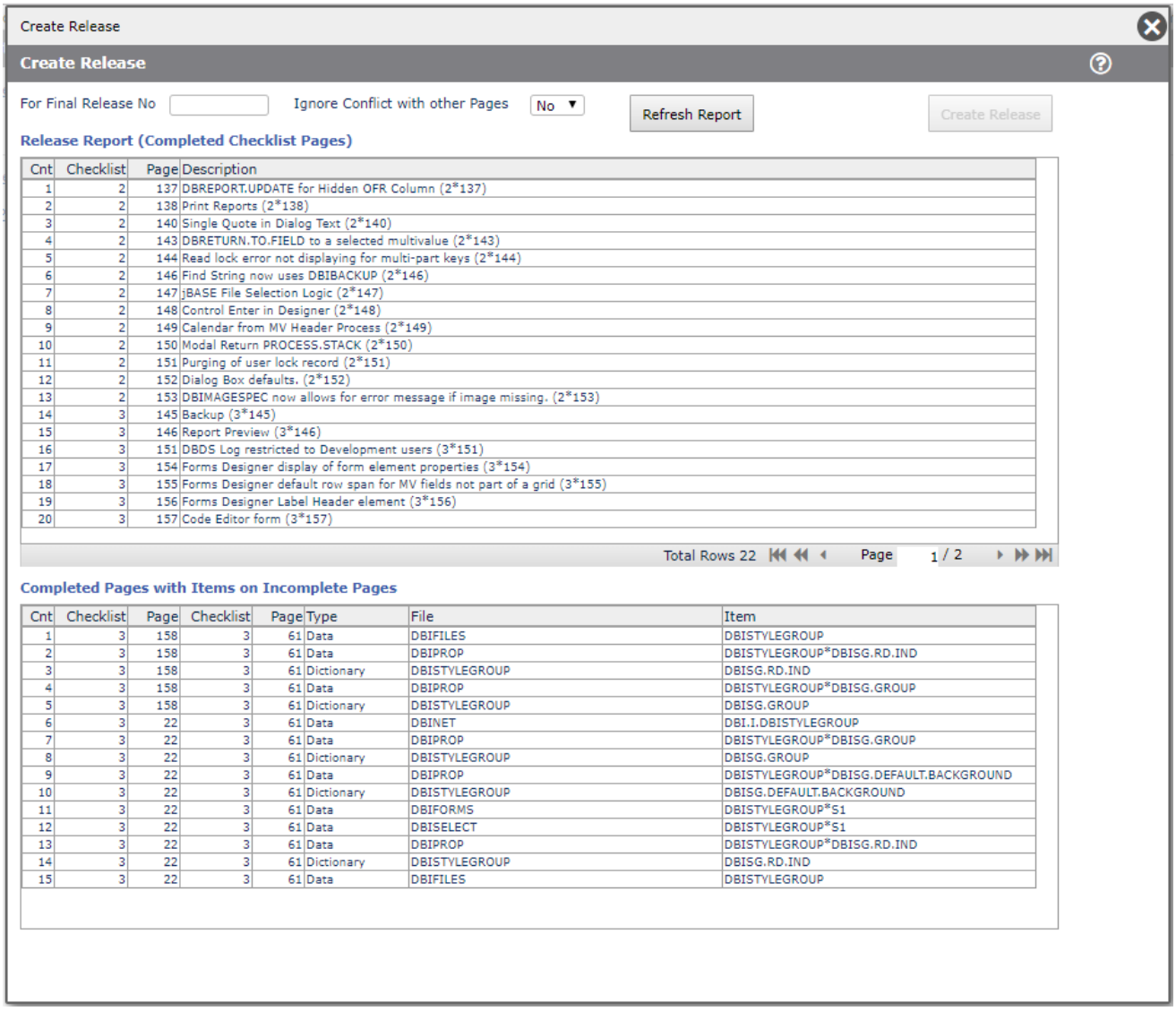
After loading a checklist transfer file a check is carried out to ensure that there are no Dictionary/Field Property mismatches. If any mismatches are found then there will be an entry in the update log as shown below.



Create Release

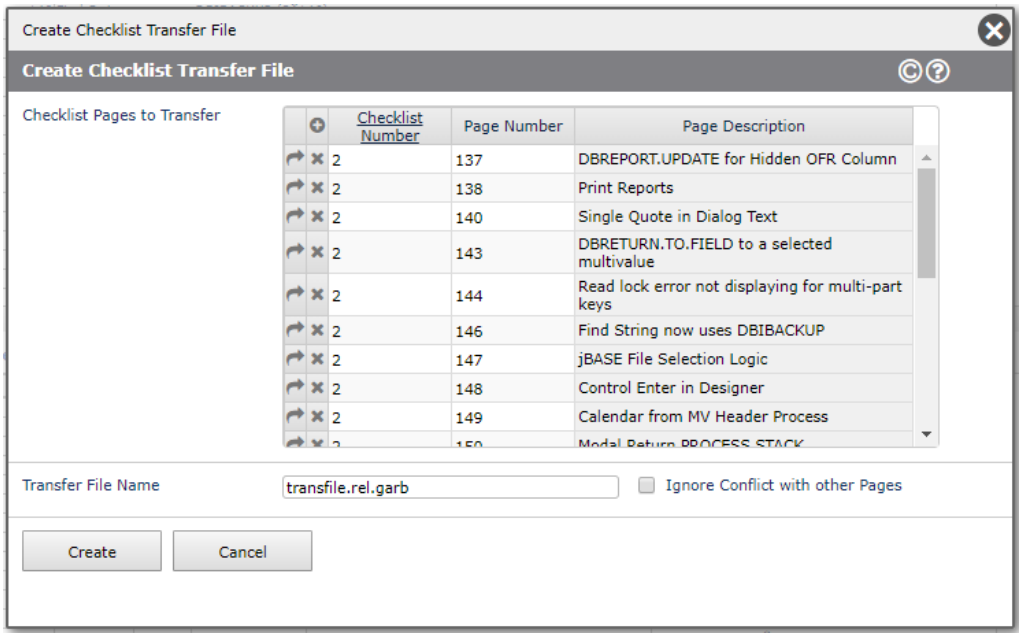
This option displays the list of completed tasks. Completed tasks have the Date Completed and Date Tested fields populated. Once again you can choose to ignore conflicts by selecting *Yes* in the *Ignore Conflict with other Pages* field.

By default completed pages with items on incomplete pages will be excluded from the release.

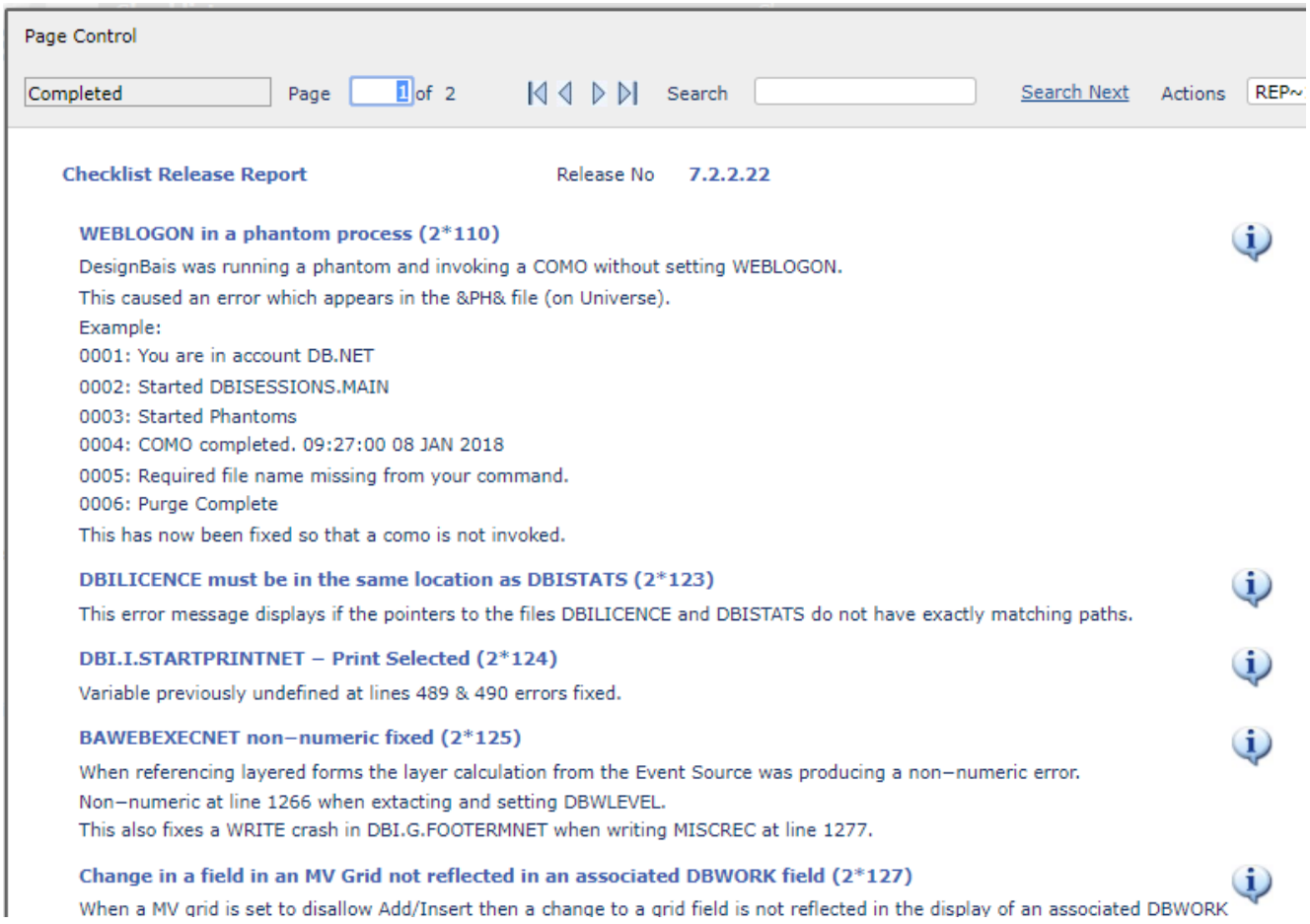


The *Create Release* button is enabled after the *For Final Release No* is entered.

Click this button to display the *Create Checklist Transfer File* with the list of checklist pages pre-populated and with the default *Transfer File Name* set to "*transfile.rel.*":WEBLOGON.

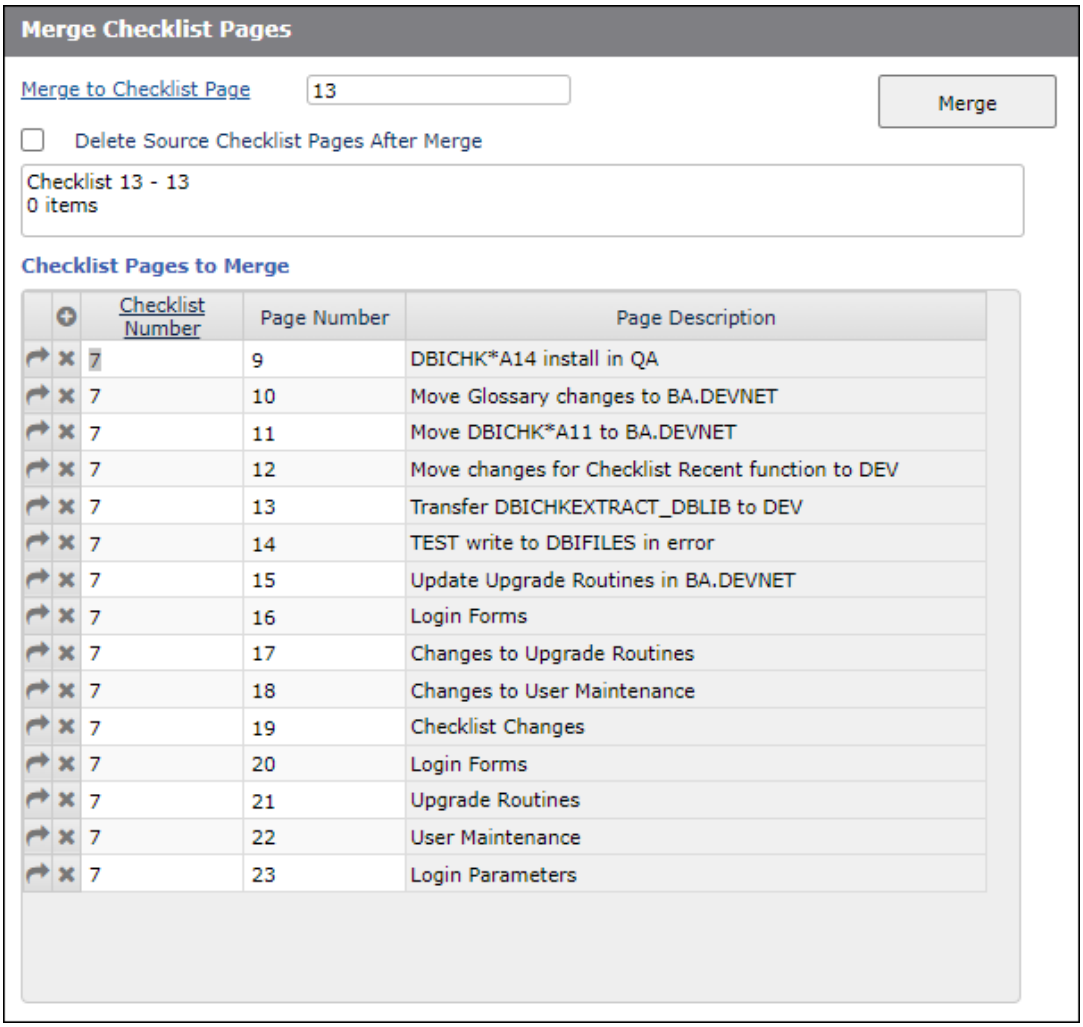


The *(Release) Report* option produces a report of the pages contained in the final release.

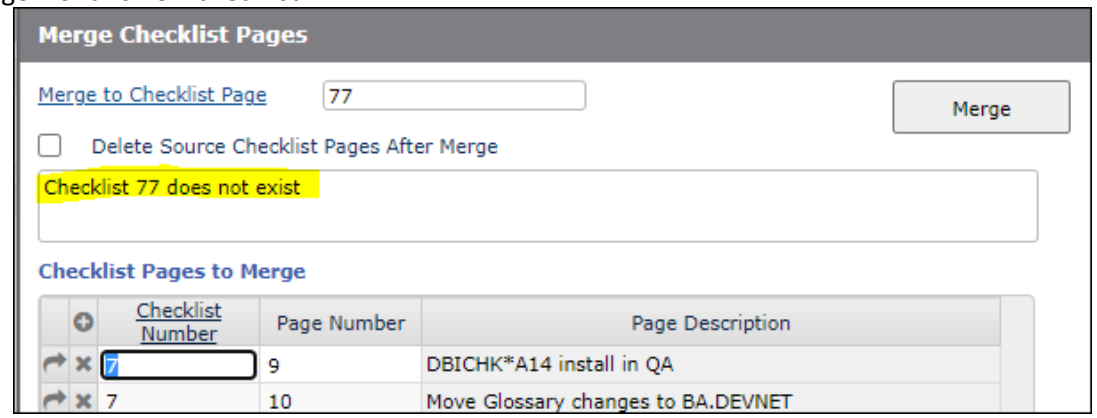


Merge Checklist Pages

This option allows you to merge selected checklist pages into an existing or a new checklist. You can specify just the target checklist id, or both the checklist and page id. In the latter case separate the checklist and page ids with an asterisk. If the page is not designated then items will be added as a new page.

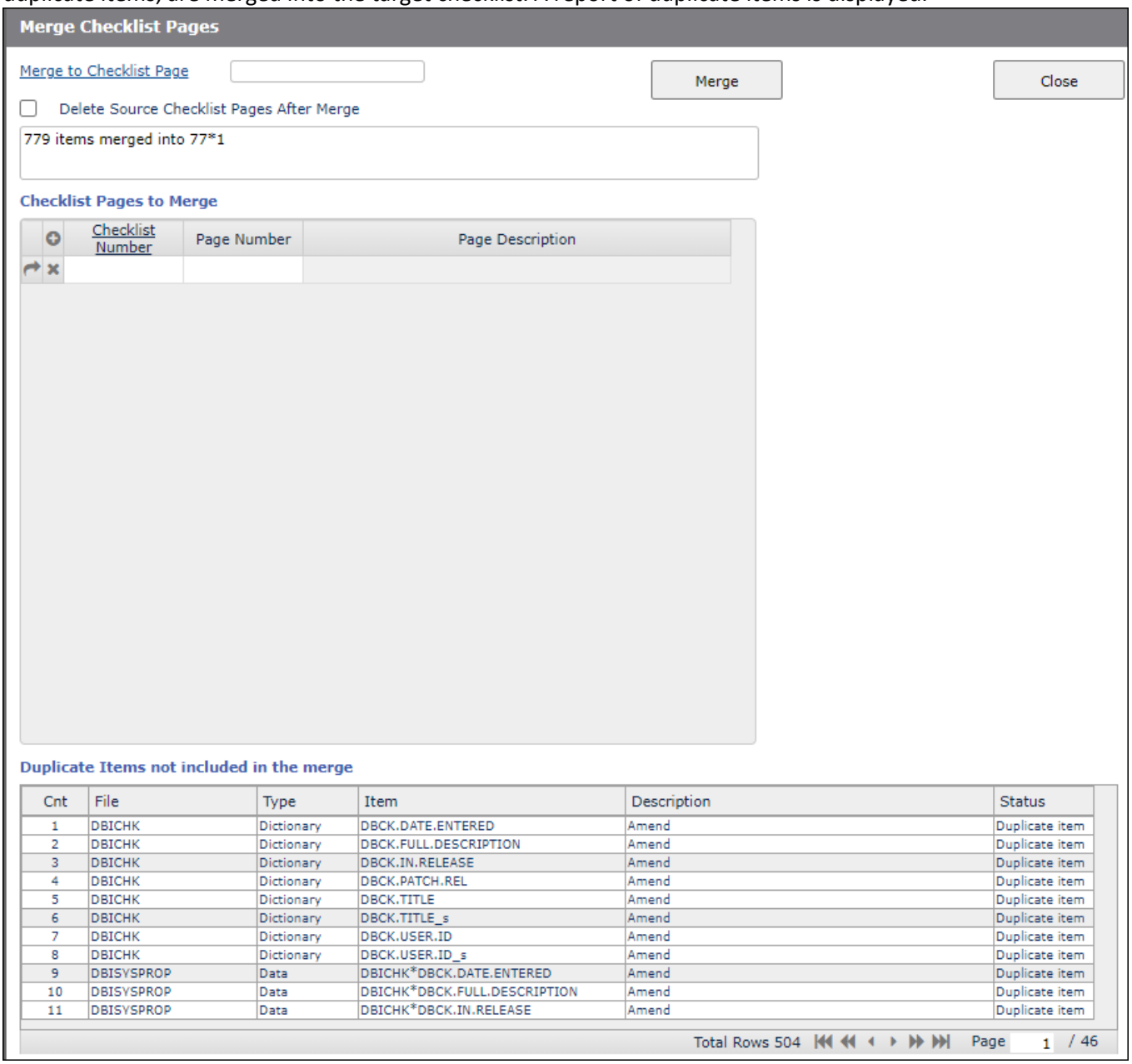


The message advises if the target checklist does not exist, in which case it will be created and all items will be loaded into page 1 of this new checklist.



Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Click the *Merge* button to initiate the merge. As shown below all items on the *Checklist Pages to Merge*, excluding duplicate items, are merged into the target checklist. A report of duplicate items is displayed.



Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Chapter 21 – End of Period

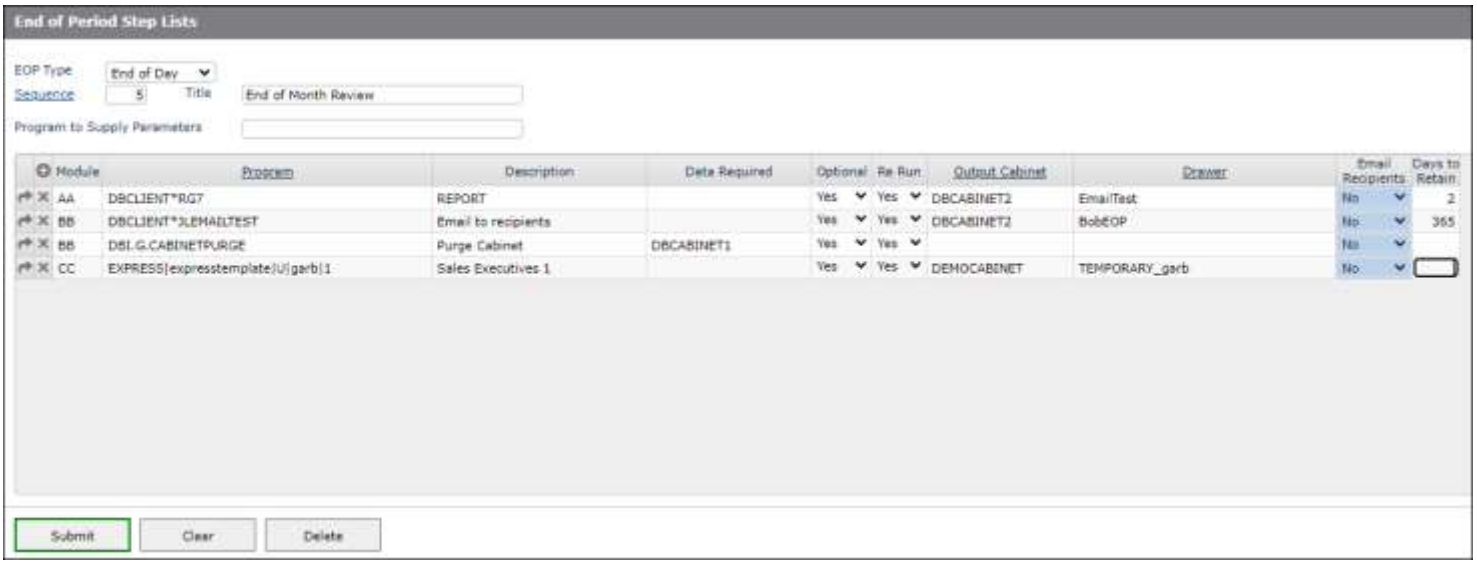
End of Period

Most applications have some form of *end of period* processing. These processes usually have multiple tasks.

The DesignBais *end of period* processing allows for the definition of the multiple tasks into distinct step lists.

End of Period Step Lists

This form provides for the set-up of these step lists.



EOP Type — You can choose to define End of Day, End of Week, End of Period or End of Year processes. Within each of these types of process multiple definitions are possible using the Sequence. Within each definition a list of mandatory or optional programs and reports can be defined along with the data needed at runtime for these programs/reports

Sequence — A field used to define multiple end of period processes of the same EOP type.

Title — A descriptive title for the end of period definition.

This defines a user written program that performs 3 functions depending upon the PROCESS.PARAMETER passed. See example program DBI.P.EOP.DATA.EXAMPLE.

PROCESS.PARAMETER<1,1> = 1

The program should optionally return a report definition in OUTPUT.REPORT(1) with a report name of R.CALENDAR. This report will be displayed on the *end of period* submission screen.

PROCESS.PARAMETER<1,1> = 2

This should return a list of valid data parameters in PROCESS.PARAMETER<1> value mark delimited

e.g. PROCESS.PARAMETER = ''
PROCESS.PARAMETER<1,-1> = '@RUNDATE'
PROCESS.PARAMETER<1,-1> = '@LASTRUN'
PROCESS.PARAMETER<1,-1> = '@PERIOD'
PROCESS.PARAMETER<1,-1> = '@YEAR'

PROCESS.PARAMETER<1,1> = 3

When called with this option PROCESS.PARAMETER<1,2> will contain the key to the end of period run record in the file DBIEOP. The parameterised input data in the end of period record in attribute DBEP.DATA should be replaced with the actual run data and the end of period record re-written.

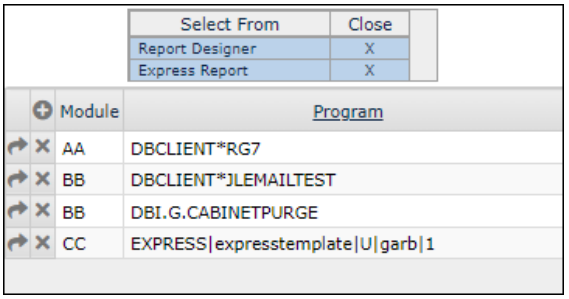
The end of period record is defined as follows

```
*************************************************************************
** Equates for file DBIEOP
*************************************************************************
   EQU DBEP.KEY                 TO      0  ;* Key
   EQU DBEP.USER                TO      1  ;* User
   EQU DBEP.DATE                TO      2  ;* Date Submitted
   EQU DBEP.TIME                TO      3  ;* Time Submitted
   EQU DBEP.DATE.TO.START       TO      4  ;* Date to Start
   EQU DBEP.TIME.TO.START       TO      5  ;* Time to Start
   EQU DBEP.START.DATE          TO      6  ;* Date Started
   EQU DBEP.START.TIME          TO      7  ;* Time Started
   EQU DBEP.FINISH.DATE         TO      8  ;* Date Finished
   EQU DBEP.FINISH.TIME         TO      9  ;* Time Finished
   EQU DBEP.STOP.USER           TO      10  ;* Termination Requested User
   EQU DBEP.STOP.DATE           TO      11  ;* Date Termination Requested
   EQU DBEP.STOP.TIME           TO      12  ;* Time Termination Requested
   EQU DBEP.TOGGLE              TO      13  ;* Stop/restart
   EQU DBEP.TYPE                TO      14  ;* Type
   EQU DBEP.TITLE               TO      15  ;* Title
   EQU DBEP.LAST.RUN            TO      16  ;* Date Last Run
   EQU DBEP.SEQ                 TO      17  ;* Seq
   EQU DBEP.REC.TYPE            TO      18  ;* Record Type
   EQU DBEP.PARAM.PROGRAM       TO      19  ;* Program to Supply Parameters
   EQU DBEP.STEP                TO      20  ;* Step Number
   EQU DBEP.PROGRAM             TO      21  ;* Program
   EQU DBEP.DESC                TO      22  ;* Description
   EQU DBEP.DATA                TO      23  ;* Data Required
   EQU DBEP.CABINET             TO      24  ;* Output Cabinet
   EQU DBEP.STEP.START.DATE     TO      25  ;* Date Started
   EQU DBEP.STEP.START.TIME     TO      26  ;* Time Started
   EQU DBEP.STEP.FINISH.DATE    TO      27  ;* Date Finished
   EQU DBEP.STEP.FINISH.TIME    TO      28  ;* Time Finished
   EQU DBEP.MESSAGE             TO      29  ;* Message
   EQU DBEP.REPORT.FILE         TO      30  ;* Report File Name
   EQU DBEP.OPTIONAL            TO      31  ;* Optional
   EQU DBEP.RUN.STEP            TO      32  ;* Run this step
   EQU DBEP.MODULE              TO      33  ;* Module
   EQU DBEP.MONITOR             TO      35  ;* Monitor Job
   EQU DBEP.LAST.REFRESH        TO      36  ;* Last Refreshed
   EQU DBEP.RUN.DATA            TO      37  ;* Data supplied
   EQU DBEP.DRAWER              TO      38  ;* Dawer
   EQU DBEP.RERUN               TO      39  ;* Re Run
   EQU DBEP.ACCOUNT             TO      40  ;* Account
```

Module — Passed to the program used to supply input data if parameterized input data is module dependant

Program — The BASIC program or DesignBais Report (Filename*Reportname) or eXpress Report to run. eXpress report names are prefixed with the string "EXPRESS|". The header contains a lookup process for Report Designer and eXpress reports.

If entering the name of an eXpress report manually (without using the lookup) you do not need to enter the "EXPRESS|expresstemplate|" string. It will be added by DesignBais.



Note that the *Run eXpress Report* option in the eXpress menu displays the names of reports that can be run by the user. Mouseover a report name to see the eXpress report id. This is the id that is required when entering values in this *Program* column.

Description — A description of the program/report to be run.

Data Required — The data required to run the program/report comma separated. If the data is variable the variable name should start with a @
e.g Y,@RUN.DATE,@PERIOD,@YEAR
This line is then replaced by the 'Program to Supply Data' at submission time.

Optional — If this step is defined as optional you can click the line at submission time to run/not run this particular step.

Re-Run — If this step can be re-run without user intervention it should be flagged here as re-run – Yes. If the end of period stops/fails during this step the job may be re-submitted and the step will be automatically re-run. If the step is not re-runnable and end of period fails during the step the step will need to be manually completed and the end of period run record updated to indicate that it has been completed before the end of period can be re-submitted

Output Cabinet — The name of the output container to be used to store reports created by this step of the end of period process. The name of an existing Cabinet must be entered. If this field is left blank then the first cabinet shown in the eXpress Cabinet Description in eXpress Setup will be used. A lookup is provided.

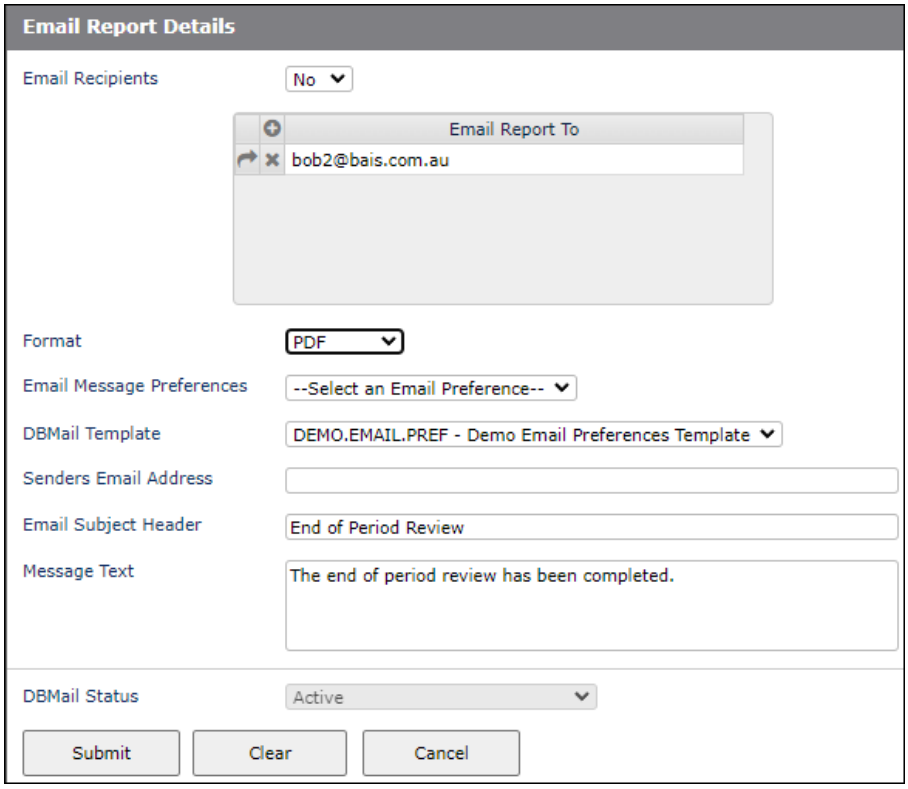
Drawer — A Cabinet drawer may be defined for the report output – if this is not defined a default drawer will be created based on the date of the run. @ variables (concatenated by a colon) may be used to construct the name. The four @ variables available for use here are @DATE, @TIME, @WEBLOGON and @DBIACCOUNT. A lookup is provided.

Email Recipients — Select Yes to use the @EMAILPRINT feature. Report pages will be automatically emailed to the specified recipient on each page. Otherwise select No. You can then, if required, enter a list of email recipients in which case the entire report will be emailed to the specified recipients in the specified format (PDF, CSV or Excel).

The name of a cabinet and drawer must be explicitly supplied.

If email recipients are specified, by either option, then the email subject, text and DBMail template can be specified in the form DBIEOP_A20 as shown in the example below:

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3



Days to Retain — The number of days to retain output in the cabinet/drawer before the output is purged.

Chapter 22 – DesignBais eXpress

DesignBais eXpress

The DesignBais eXpress module is designed to provide developers and end-users with two distinct tools that significantly improve access to reports and information stored in your application database. The two tools are:

eXpress Reporting

Provides end-users with an ad-hoc reporting tool that is able to navigate your application data structure. The graphical nature of the interface makes it very easy for end-users to interrogate the application tables.

DesignBais security is applied within this tool, which ensures that end-users are provided with only the information that they have access to.

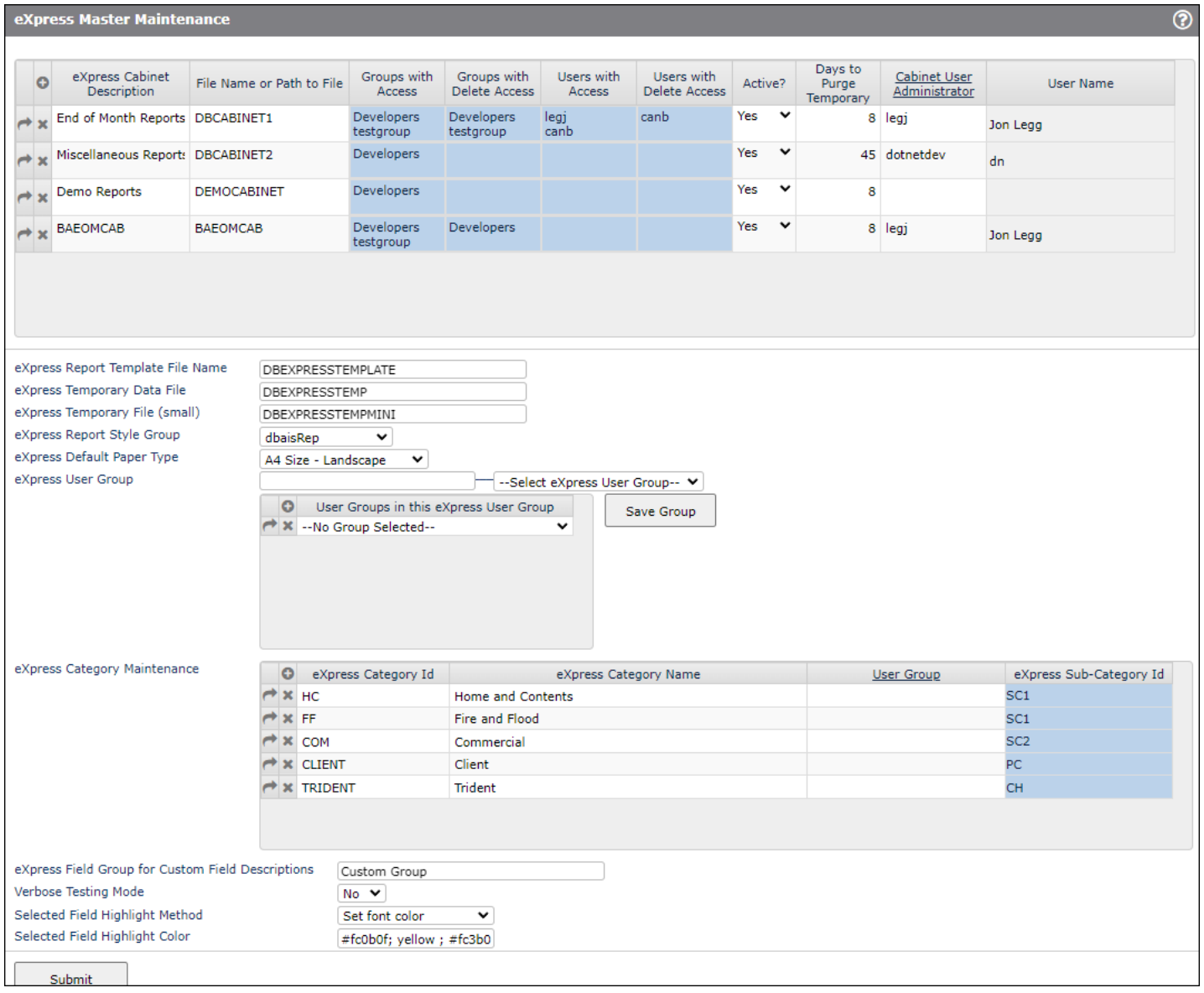
There are six menu options in the eXpress module. Each of these options can be placed within your own menu structure if required.

Before you start using the DesignBais eXpress module there are some parameters to be set-up. Refer to the eXpress Setup option below.

eXpress Setup

The eXpress setup form is used to define files that are to be used for the various features of the DesignBais eXpress module and to define eXpress Groups.

This option displays the following form.



Prompts:

eXpress Cabinet Description The cabinet description will be viewable by end-users. If is therefore important to provide a meaningful description. Eg. End of Month Reports, Product Reports.

File Name or Path to File This is the physical filename to be used by cabinets. The file must exist as DesignBais will not create the file for the cabinet reports to reside in.

It is advised to make these cabinet files reasonably large and also ensure that each group has a large frame count (separation) as reports can be large individual records. DesignBais limits each output block to a maximum of ten pages.

Create Cabinet files as directory type files (type 19 in UniVerse).

You must not create a directory type file in UniData on Windows because this platform does not allow the use of certain characters in file or directory names, including the pipe (|) character. DesignBais uses the pipe character in the record id of each cabinet drawer.

Groups with Access Assign a list of user groups that have access to the cabinet.

Groups with Delete Access Assign a list of user groups that have access to the cabinet.

Users with Access Assign a list of users that have access to the cabinet.
If both access fields are left blank, all users have access to the cabinet.

Active If this flag is set to "No", it will no longer appear in the Cabinet views.

Days to Purge Temporary Each temporary drawer will have old reports purged regularly. The default is 8 days. You may change this value.

eXpress Report Template File Name

For eXpress to function, a filename needs to be created to store the report skeletons produced using the eXpress interface. The filename entered at this prompt must be valid and able to be opened within the current account.

eXpress Temporary Data File The file defined in this field will be used by eXpress to store temporary report data. The eXpress interface ensure that the data is cleaned up immediately after the completion of the report, so it is not necessary to make this file very large. A file name must be nominated for eXpress to function.

eXpress Temporary File (Small) This file controls the User Interface in the eXpress report. It is only designed to have a very small number of records. This file should only be about 100 frames in total. Any size larger than that will detract from the user interface.

eXpress Report Style Group The stylegroup assigned in this field will be the style used for eXpress reporting.

eXpress Default Paper Type This field will set the default paper size for eXpress reports. Users are able to change the default when building a report.

eXpress Group Enter a code to designate an eXpress Group. Express Groups are a collection of User Groups. eXpress reports can be assigned for access by a single User Group or a single eXpress Group. By using an eXpress Group it is possible to allow access to a report by multiple User Groups. The eXpress Group code should be short and cannot be the same as any existing user Group codes.

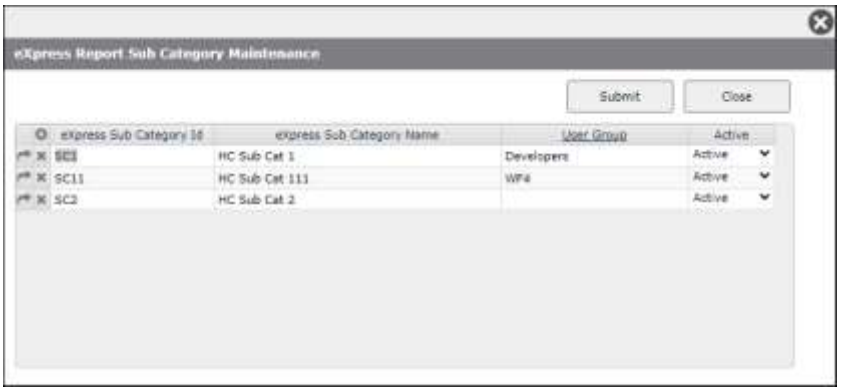
eXpress Group User Groups Enter the list of User Groups that are to be included in the eXpress Group.

eXpress Category Maintenance - eXpress Category Id

This field contains an eXpress Category code to which a field name (description) can be associated for use in eXpress reports. This
name is used to provide an alternative field description for the eXpress Category selected for an eXpress report.

eXpress Category Name The name or description of the eXpress Category.

User Group The user group that has access to eXpress reports created for this eXpress Category.



eXpress Sub-Category Id eXpress Sub Category determines the description of fields used in eXpress reports. Based on the sub category a field can have varying descriptions. The Sub Category Id can be any short code of your choice. You can then associate a name or description with the code.
When creating an eXpress report you can determine the description of the available fields for the file of the report by selecting an appropriate sub category.

eXpress Sub-Category Name The name or description of the eXpress Sub Category.

User Group The user group that has access to eXpress reports created for this eXpress Sub-Category.

Active The eXpress Sub Category status may be active or inactive. Set active status off if the sub category is not to appear in the list of available sub categories for an eXpress report. Users can still execute existing eXpress reports linked to inactive sub categories.

eXpress Field Group for custom Field Descriptions

The eXpress Field Group name to be used for custom field descriptions. When custom field descriptions are created for an eXpress Category / Sub-Category the field property will be placed in an eXpress Field Group with this name. If this field is left blank then a field property with a custom field description will remain in its currently assigned eXpress field group.

Verbose Testing Mode This setting only effects users who are members of the Developers or DBAdministrator Groups. Set this option to 'Yes' to invoke verbose mode for eXpress Reports . The user como and the phantom output (&PH& for UniVerse) display variables to allow the developer to track the display of fields on an eXpress report.

Selected Field Highlight Method

Select how to highlight field names, in the left hand side field name list, that have been selected for use in the eXpress report being created. Set background will use the color entered in the next field to change the background color of the field name. Set font will change the font color.

Selected Field Highlight Color The color to use for highlighting fields selected for inclusion in the eXpress report. The color will be applied as font color or background color based on the Selected Field Highlight Method setting. Two colors can be entered separated by a semicolon (;) in which case the second color will be used when the highlight method is to set background color.

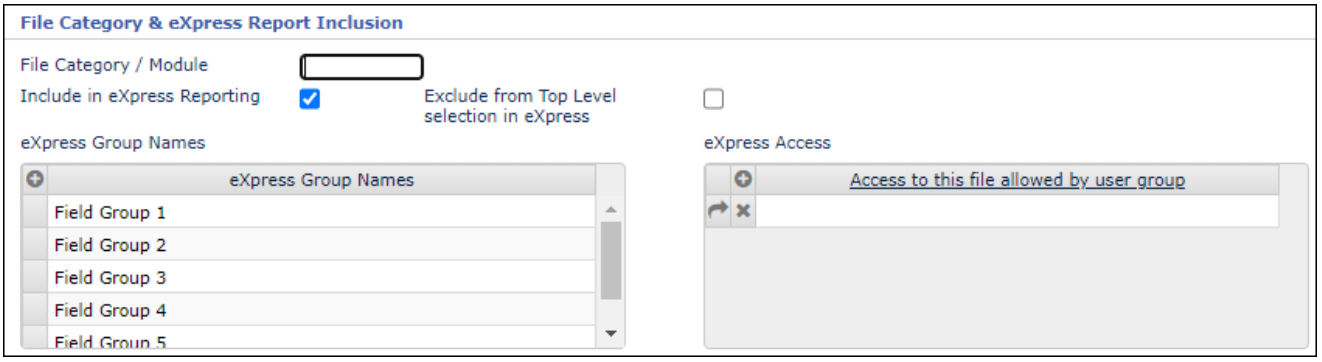
Buttons:

Submit Submit the eXpress parameters.

Error Messages:

At least one file must be set up to use Express Reports.

In the File Properties option enter the name of a file that is to be used in Express Reports. Check the field *Include in eXpress Reporting* and click the *Submit* button.



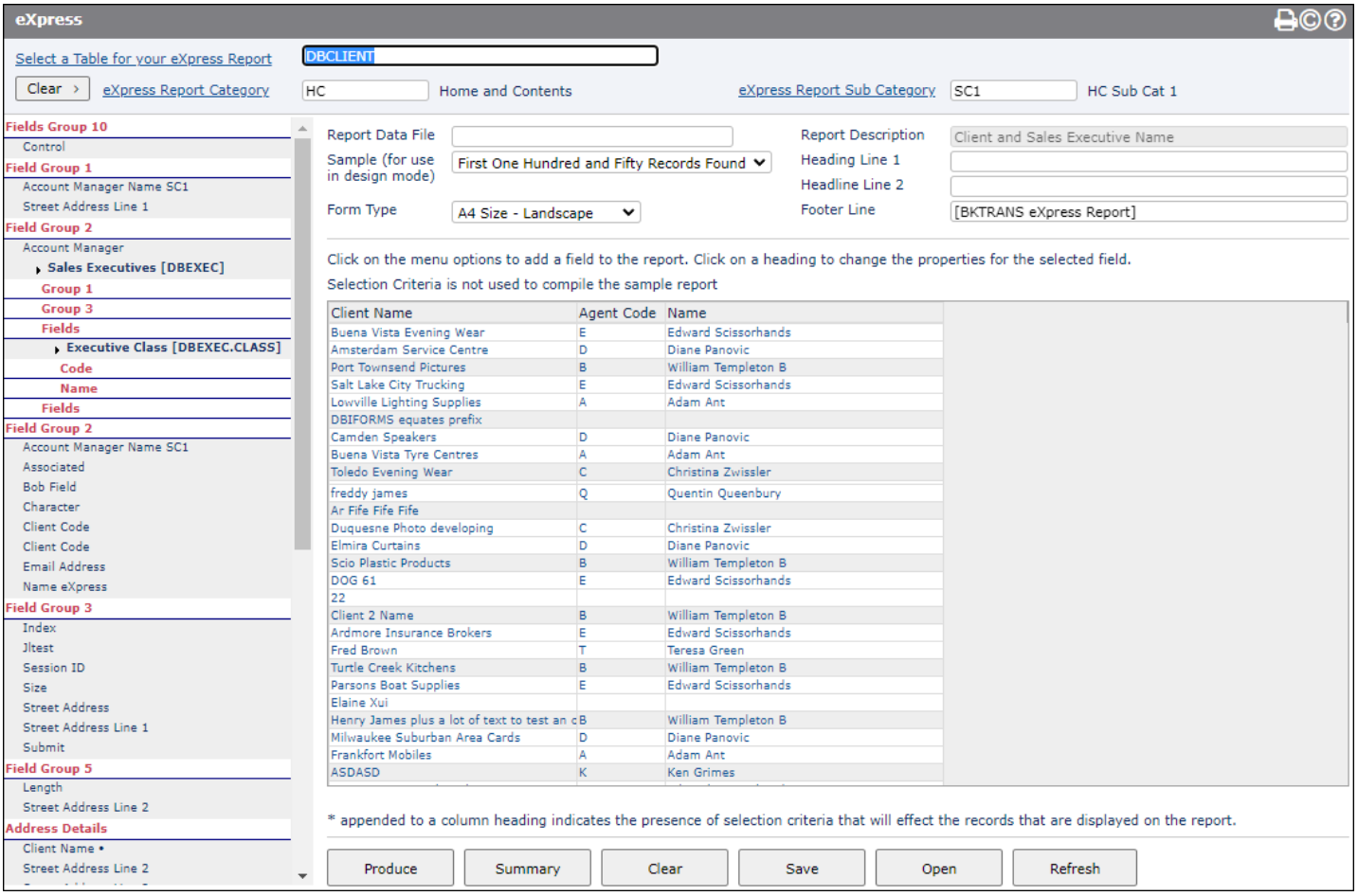
If no File Properties records have the *Include in eXpress Reporting* field checked then the following error message is displayed:

'There are no files assigned for Mini Reports. Please check with your system administrator.'

The eXpress reporting tool provides a powerful, yet simple method of extracting information from an application database. The interface is designed to be used by non-technical users.

The first screen presented to the user is the main eXpress form.

Example:



Select a Table for your eXpress Report

Tables that have been flagged in File Properties as available for eXpress Reporting can be displayed and selected by clicking the hyperlink lookup button Select a Table for your eXpress Report.

Select the table that contains the reporting field definitions for the data that is required in the eXpress Report.

eXpress Report Category

Click the button to display the list of available categories.

eXpress Report Sub-Category

Click the button to display the list of available sub-categories.

Categories and Sub-Categories are setup in the *eXpress Setup* form.

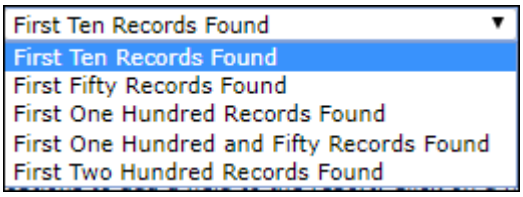
The eXpress Category code that is assigned to an eXpress report is used to provide an alternative field description for the report. The Category Id is a code with an associated name. The Sub-Category Id can be any short code of your choice. You can then associate a name or description with the code.

The eXpress Category and Sub-Category determine who can view the report and the particular description for fields used in an eXpress reports. Based on the sub-category a field can have varying descriptions.

When creating an eXpress report you can determine the description of the available fields for the file of the report by selecting an appropriate eXpress Category and Sub-Category. These codes can determine the description of fields used in eXpress reports.

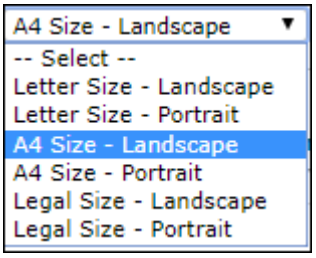
Report Data File Use this option to define the name of a file that will be used to to derive the data for a report that is based on another table on which the Field Definitions (Dictionary Items) are defined.

Sample (for use in Design mode)



By default the DesignBais eXpress sample engine finds and displays the first ten records available to the end user. These records have passed any security requirements before being displayed on the sample report. If a greater sample of the table is required, the user has the following choices.

Form Type Select the form type required for the report. This will identify the size of the paper to be used for the report and also the orientation.



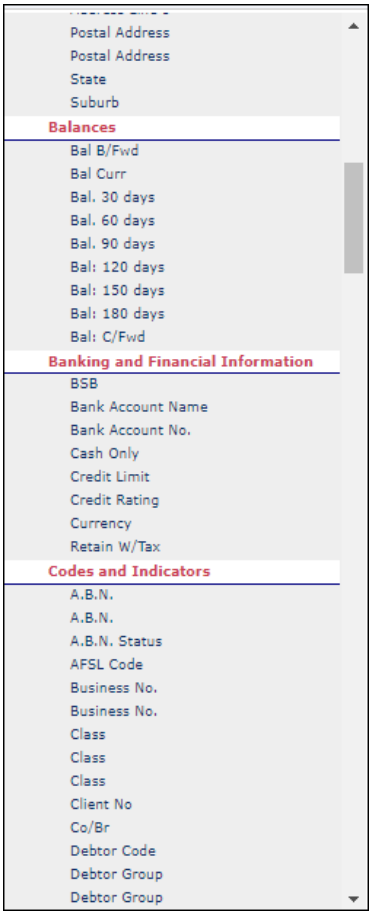
Report Description The Report Description is displayed here when an existing saved report is recalled.

Heading Line 1 The main heading to appear at the top of the report.

Heading Line 2 The secondary heading line.

Footer Line Each report has a small footer area assigned. If you wish text to appear in this section, enter the desired text in this field. By default the footer line is populated with the name of the table on which the eXpress report is based.

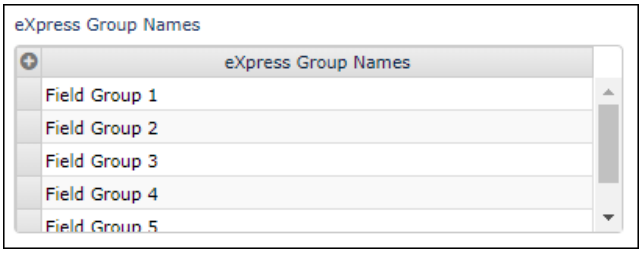
Adding fields to an eXpress report



Once a table has been selected it is simply a matter of clicking on the side menu field name to add it to the sample report. Clicking the same field again will remove it from the report.

After selecting a table the list of available reporting fields is displayed down the left-hand side of the form. Fields can be grouped under defined headings (shown in red) in order to facilitate the selection of fields by users less familiar with the data. These groups are called eXpress Group Names. Grouping can be set up in Field Properties for an individual field.

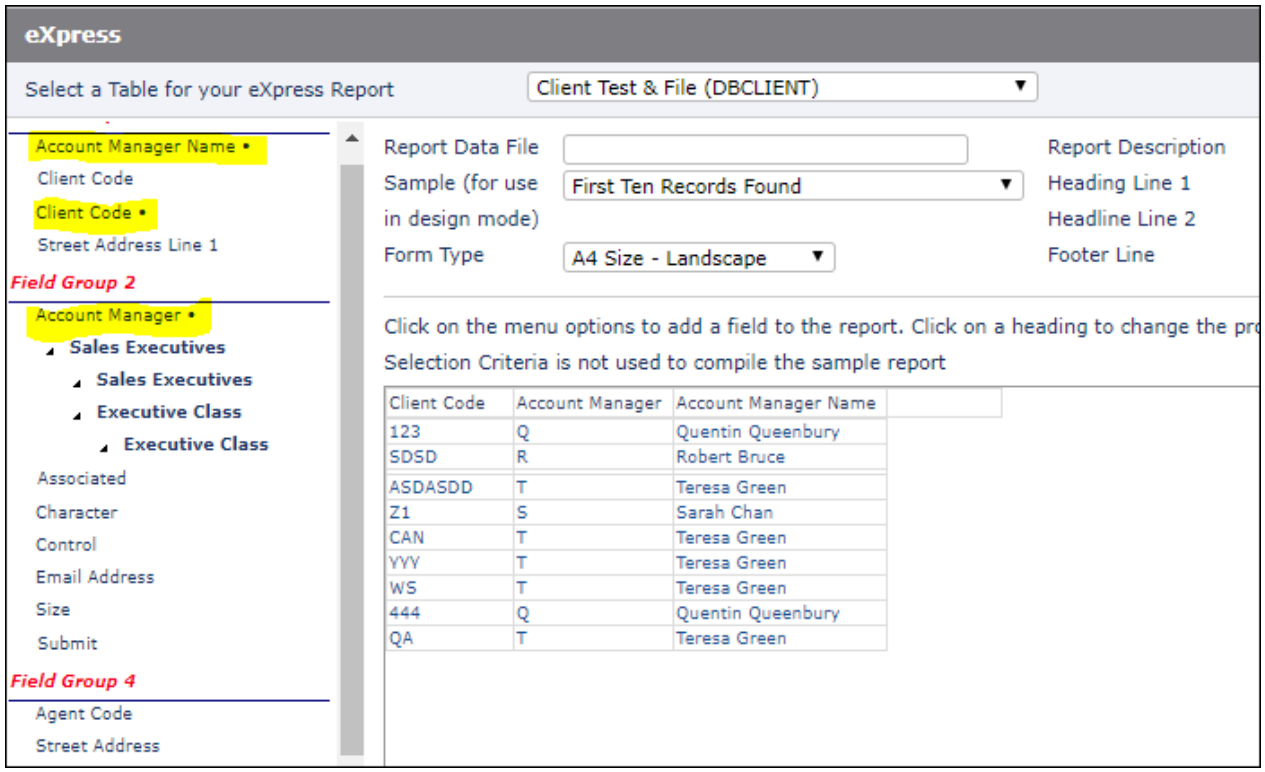
The list of eXpress Group Names for a file (table) can be created in the File Properties form for the file (table). Refer to File Properties in the Reference Manual.



In Field Properties an individual field can be assigned to one of these groups.

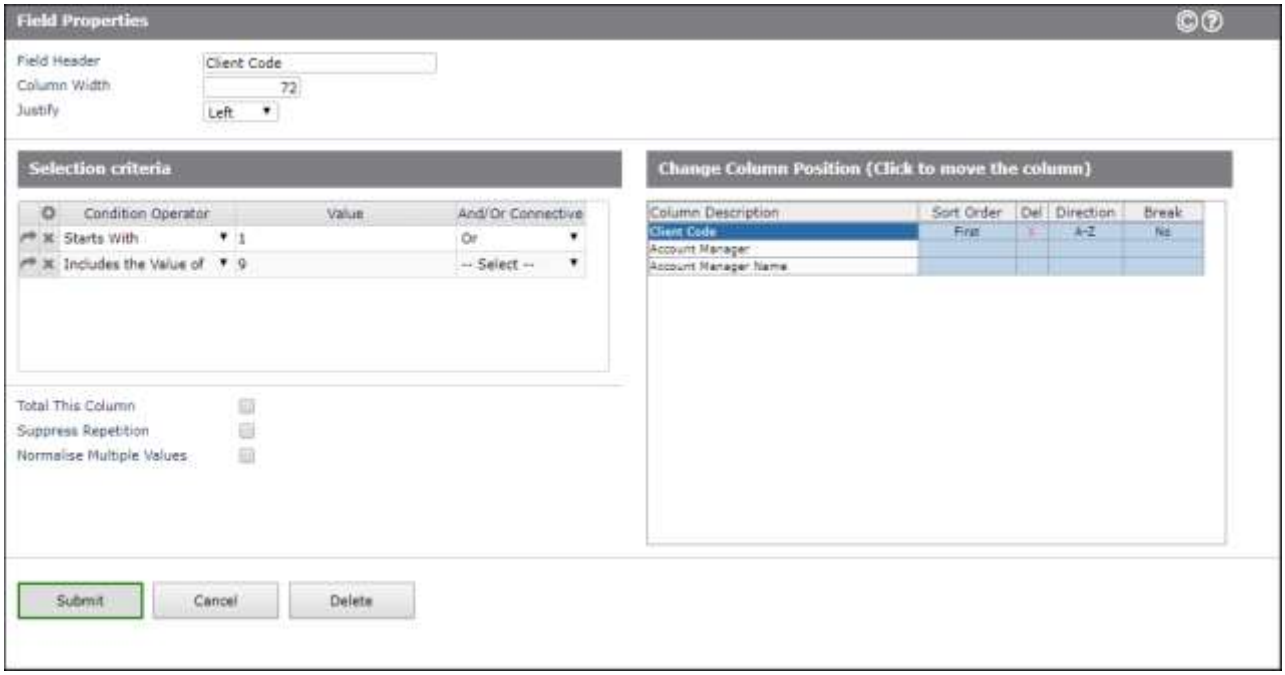
An alternative way to assign fields to eXpress Groups is to use the ***Set eXpress Field Group*** menu option.

In the example below the three fields highlighted in yellow have been placed on the report. Note that an asterisk is placed after the name of each field in the list on the left that has been added to the report.



Modifying the attributes of a field on an eXpress report

To change the display attributes of fields on the report, such as sort order and position, click on the header title of a column.



Clicking on the header row of the Client Code column opens the Field Properties form for the report field. The Field Header, Column Width and Justification can be amended.

To change the column position of the selected field, in this example we have selected *Client Code*, simply click on the row in the *Column Description* column that corresponds to the desired position. Clicking on the third row will set the *Client Code* field to be the third column of the report.

Total This Column If a numeric field is to be totaled check the *Total this column* check box. By default any numeric field that is added a report will generate a report total figure.

Suppress Repetition Suppress Repetition a can be checked in order to display the field on the initial row of a record on a report and to then suppress the display for all rows that display multiple values of other associated fields.

Normalise Multiple Values

If a single field has multiple values (multi-value), the standard behaviour when exporting the report to a spreadsheet is to flatten these multi-values and create more columns in the spreadsheet. Tick this check box to override this bevahiour, then. This will then duplicate the rows of all other non multi-value columns for each individual value.

Sort Order By default, values in columns are presented in the order that they are found on the database table. To provide better organisation on a report, it may be necessary to sort the report by the values that appear in certain columns.

In the column *Sort Order* click the row corresponding to the field that you want to be the primary sort field. The *Sort Order* row will display *First*. Click other rows as required to define secondary sort fields. The row will display *Second*, *Third* etc to indicate the field sort order.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Del Click the *Del* column to remove a specified sort order definition for the field on the row that is clicked. Other defined sort order fields will be automatically adjusted.

Direction Click rows in this column to define ascending and descending sort order.

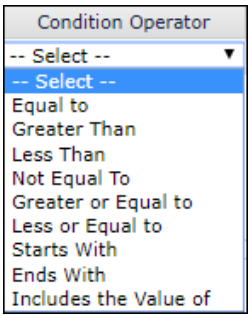
Break It may be necessary to add break totals when a value in a column changes. This should only be used when a column is sorted. Select the type of break if required. The options are *No* to suppress the break for the field on this row, *Yes* to break and *Yes+Page* to effect a new page on the break.

Selection Criteria It may necessary to specify a selection criteria for a column. In doing this, you are specifying that only the nominated values are to appear in the column.

Condition Operator

The Conditional Operator specifies how to test the values in the column.

Select the conditional operator that will be used.



For example to select all records in the table with name = "Smith", the *equal sign* is the conditional operator. To select all records in the table with net_price >= 25.00, the *greater than, equal sign* is the conditional operator.

You may have more that one conditional operator and value in your selection criteria by using the and/or connective.

Value Enter the value that is to be searched for in the data from the table that is being reported.

And/Or Connective

The *And/Or Connective* specifies how the component rows of the selection are joined. If the selection criteria are conjunctive then select *And* otherwise select *Or*.

The selection criteria **do not** affect the sample report so as to avoid performance issues when selecting the records that match the criteria. The specified selection criteria will be applied when the user clicks the *Produce* button to run the report.

Producing the Final Report

Click the *Produce* to produce the final report.

If Selection Criteria have not been specified for any fields on the report a warning will be displayed. This serves only to allow you the opportunity to apply selection criteria, if you wish, to in order to minimise the time it takes to produce the report.

Report Preview Window

The report preview window is used to view a report that is run from eXpress or previewed from the DesignBais report generator.

The top heading row of the Preview Window contains the following fields, left to right:

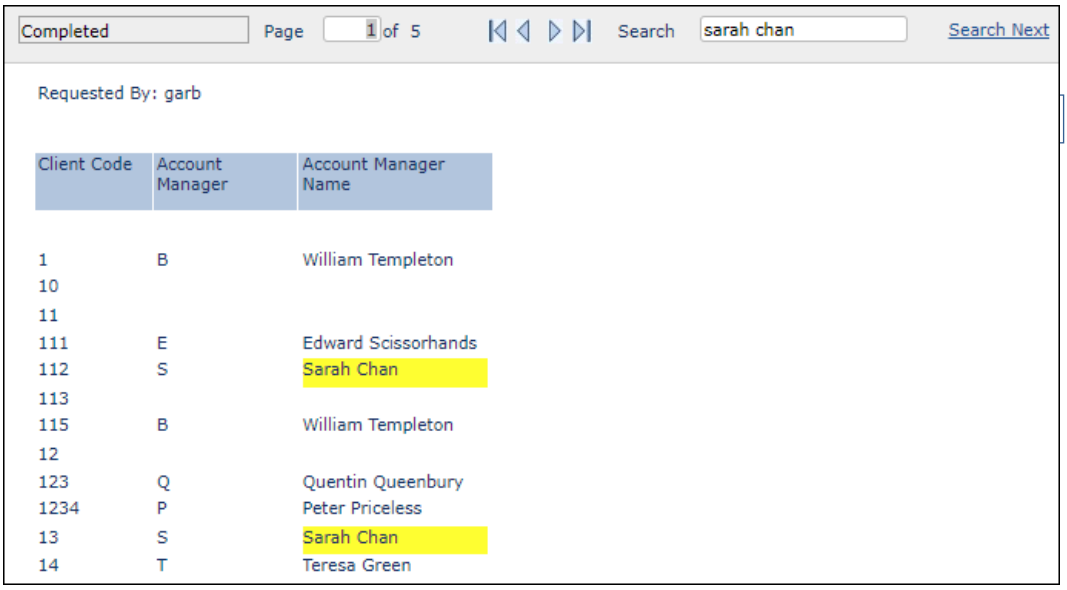
Report Status This will display *Completed* when the report is complete. Up to that time it displays a progress bar.

Page Displays the currently displayed page number and the total number of pages.

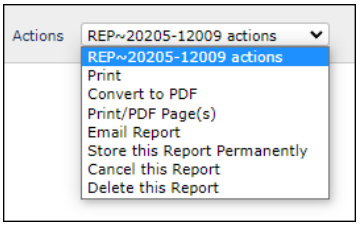
Page Control Use the arrows to move between pages or to the first or last page.

Search Search for a text string within the report.

Search Next Click to repeat the search for the specified text.



Actions The Id of the report is displayed on the top row. On a Universe database this is the id of a record in the &PH& file. The record will contain messages generated during the production of the report.

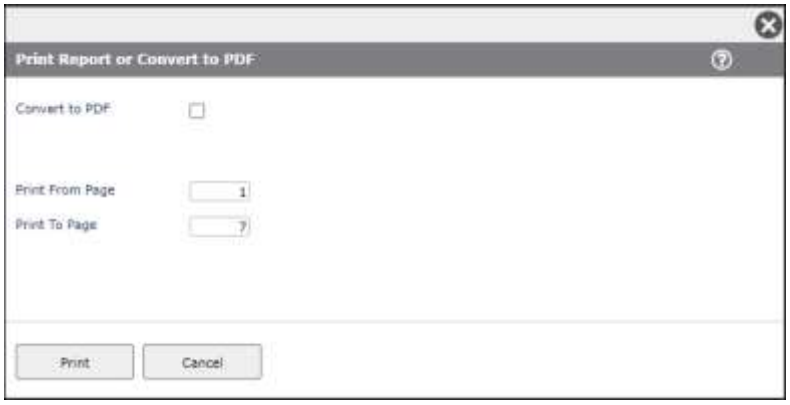


Print Print the Report

Convert to PDF Convert the report to a PDF file. See example below of the PDF output.

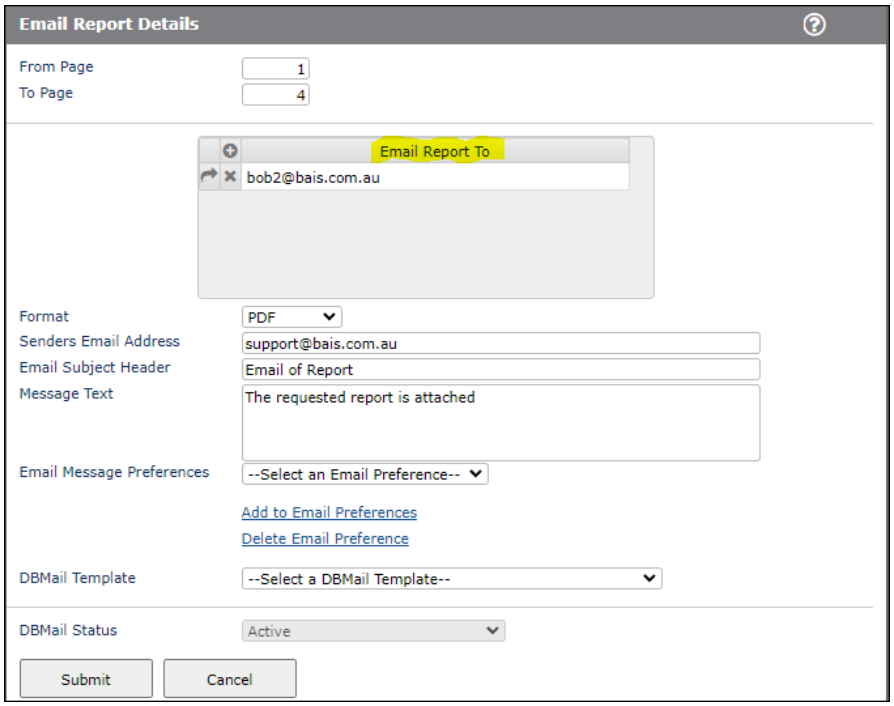
Print/PDF Page(s)

Print selected pages of the report or the PDF file.



Email Report

Email the report to a specified email address. The default is the email address attached to the user record of the user running this report.



Store this Report Permanently

Store the report in a cabinet drawer. Refer to the *Cabinet* section of this manual.

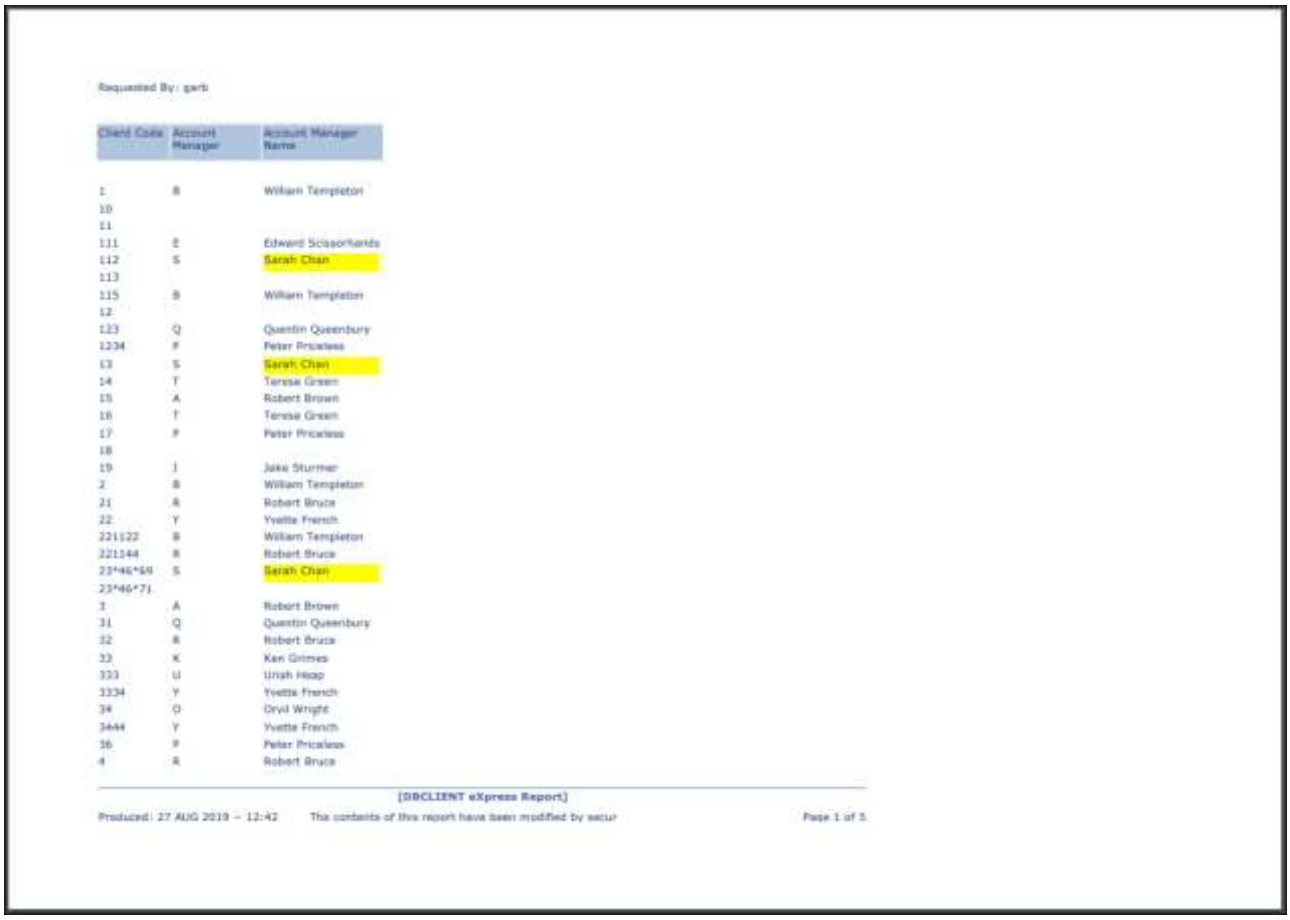
Cancel this Report

Select this option to cancel the report. This is the same as clicking the *Cancel* button in the header of form M32.

Delete this Report

This instance of the report is deleted. If viewing from a cabinet, the report is removed from the cabinet.

Example of PDF report. Note that the default footer contains the name of the file from which the report data has been extracted.



Requested By: garb

Client Code	Account Manager	Account Manager Name
1	B	William Templeton
10		
11		
111	E	Edward Scissorhands
112	S	Sarah Chan
113		
115	B	William Templeton
12		
123	Q	Quentin Queenbury
1234	P	Peter Priceless
13	S	Sarah Chan
14	T	Teresa Green
15	A	Robert Brown
16	T	Teresa Green
17	P	Peter Priceless
18		
19	J	Jake Sturmer
2	B	William Templeton
21	A	Robert Bruce
22	Y	Yvette French
221122	B	William Templeton
221144	R	Robert Bruce
23*46*58	S	Sarah Chan
23*46*71		
3	A	Robert Brown
31	Q	Quentin Queenbury
32	R	Robert Bruce
33	K	Ken Grimes
333	U	Uriah Heap
3334	Y	Yvette French
34	O	Orvil Wright
3444	Y	Yvette French
36	P	Peter Priceless
4	R	Robert Bruce

[DBCLIENT eXpress Report]

Produced: 27 AUG 2019 - 12:42 The contents of this report have been modified by secur Page 1 of 3

XLS Convert the report to an eXcel spreadsheet.

Cancel Cancel the report review display and return to eXpress.

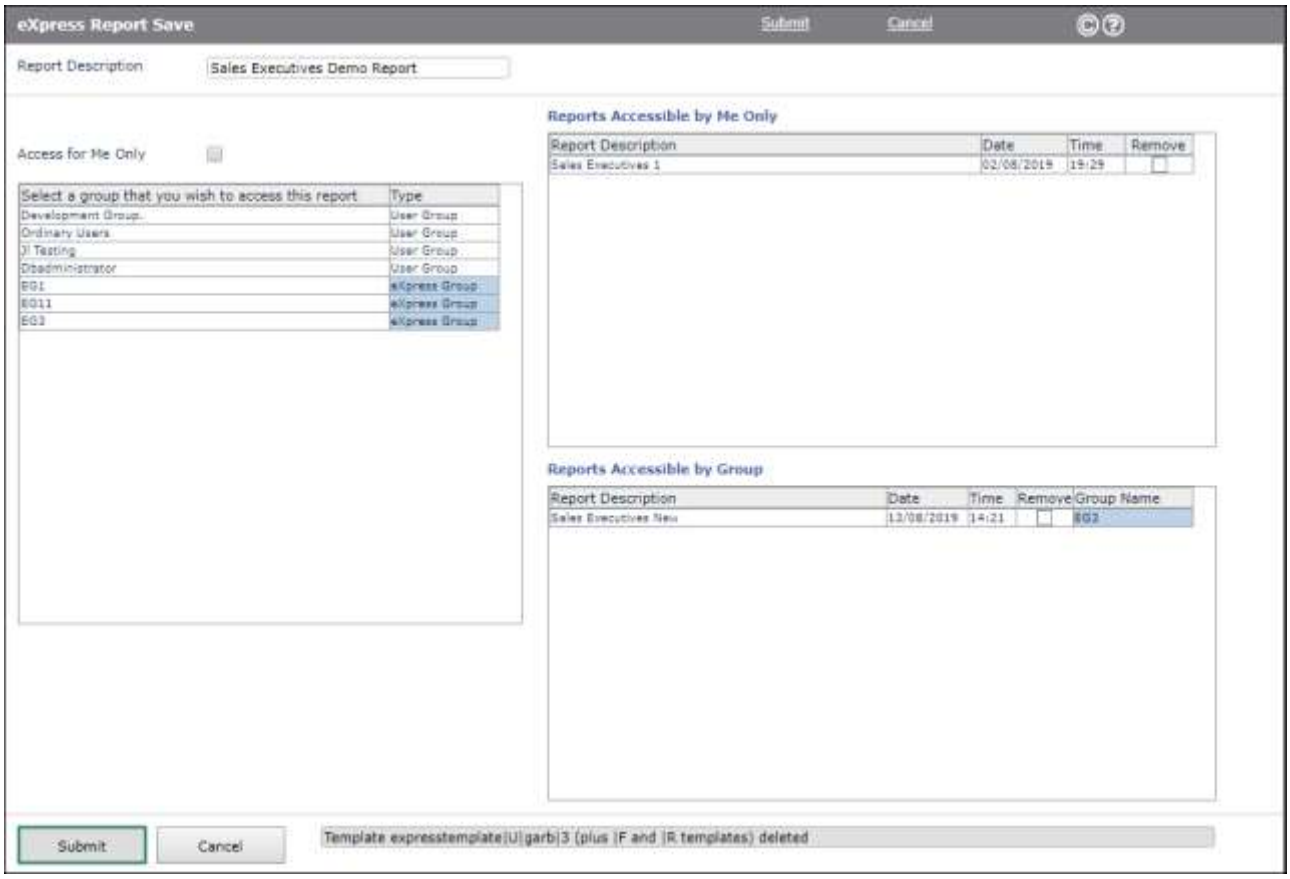
Saving an eXpress Report

Click the *Save* button on the *eXpress* form.

Report Description Enter the description to be used as the name of the report.

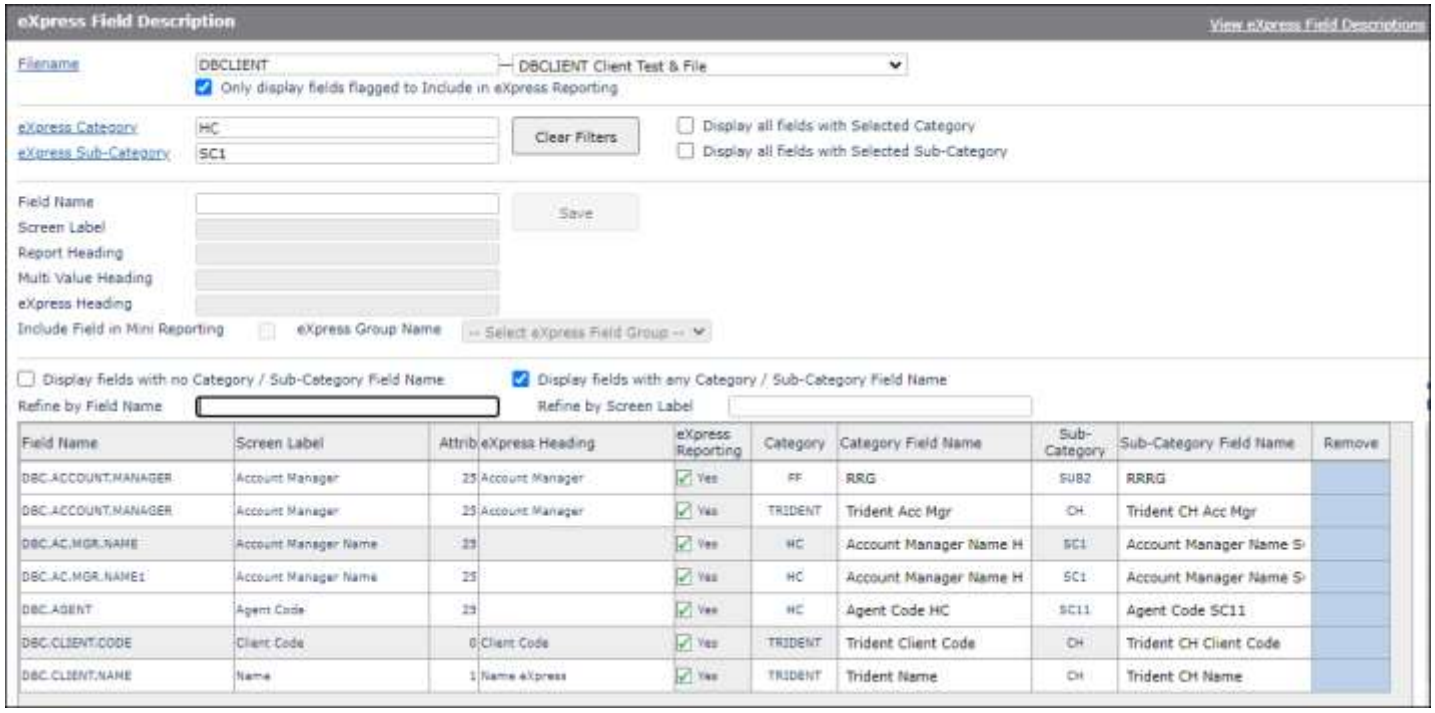
Access for Me Only By default the *Access for Me Only* check box will be ticked. After clicking Submit the name of the saved report will display in the top section titled *Reports Accessible by Me Only*.

Uncheck this check box if the report is to be made accessible to other users based on a user's membership of a User Group or eXpress Group.



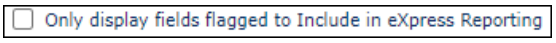
eXpress Field Description

Use this form to define alternative field descriptions for a field based on the eXpress Category and Sub-Category assigned to the field.



Enter a file name and category and sub-category codes.

The list of field properties displayed by default are those that have been flagged to be included in eXpress reporting.

Uncheck the checkbox  in order to display fields that have not been flagged to be included in eXpress reporting. Such fields will be flagged to be included in eXpress reporting if an alternative description is entered.

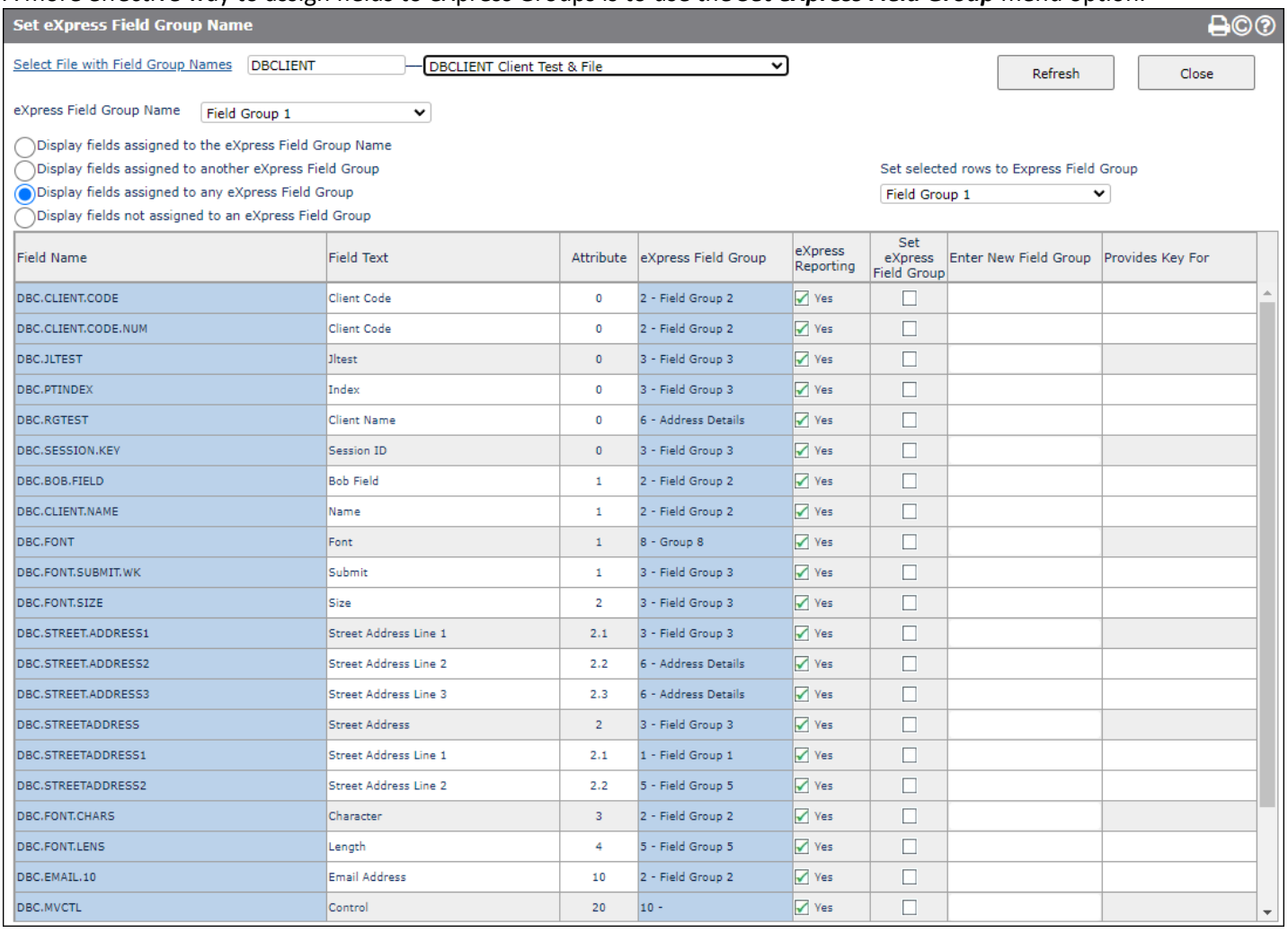
Use the 

checkboxes and refine fields to control the display of fields to which a field description can be applied. Where a description has already been entered it will display and can be amended.

You may click the *Field Name* in column 1 to allow the field properties *Screen Label, Report Heading, Multi Value Heading* and *eXpress Heading* to be amended without the need to open the *Field Properties* form. After editing these fields click the *Save* button to confirm your changes.

Set eXpress Field Group

A more effective way to assign fields to eXpress Groups is to use the ***Set eXpress Field Group*** menu option.



Field Name	Field Text	Attribute	eXpress Field Group	eXpress Reporting	Set eXpress Field Group	Enter New Field Group	Provides Key For
DBC.CLIENT.CODE	Client Code	0	2 - Field Group 2	Yes			
DBC.CLIENT.CODE.NUM	Client Code	0	2 - Field Group 2	Yes			
DBC.JLTEST	Jltest	0	3 - Field Group 3	Yes			
DBC.PTINDEX	Index	0	3 - Field Group 3	Yes			
DBC.RGTEST	Client Name	0	6 - Address Details	Yes			
DBC.SESSION.KEY	Session ID	0	3 - Field Group 3	Yes			
DBC.BOB.FIELD	Bob Field	1	2 - Field Group 2	Yes			
DBC.CLIENT.NAME	Name	1	2 - Field Group 2	Yes			
DBC.FONT	Font	1	8 - Group 8	Yes			
DBC.FONT.SUBMIT.WK	Submit	1	3 - Field Group 3	Yes			
DBC.FONT.SIZE	Size	2	3 - Field Group 3	Yes			
DBC.STREET.ADDRESS1	Street Address Line 1	2.1	3 - Field Group 3	Yes			
DBC.STREET.ADDRESS2	Street Address Line 2	2.2	6 - Address Details	Yes			
DBC.STREET.ADDRESS3	Street Address Line 3	2.3	6 - Address Details	Yes			
DBC.STREETADDRESS	Street Address	2	3 - Field Group 3	Yes			
DBC.STREETADDRESS1	Street Address Line 1	2.1	1 - Field Group 1	Yes			
DBC.STREETADDRESS2	Street Address Line 2	2.2	5 - Field Group 5	Yes			
DBC.FONT.CHARS	Character	3	2 - Field Group 2	Yes			
DBC.FONT.LENS	Length	4	5 - Field Group 5	Yes			
DBC.EMAIL.10	Email Address	10	2 - Field Group 2	Yes			
DBC.MVCTL	Control	20	10 -	Yes			

This form allows all fields to be viewed and assigned to existing eXpress Groups, or a new eXpress Group can be entered and the File Properties list will be automatically updated.

Re-Assign Field Group

Use this option to re-assign eXpress Field Group names.

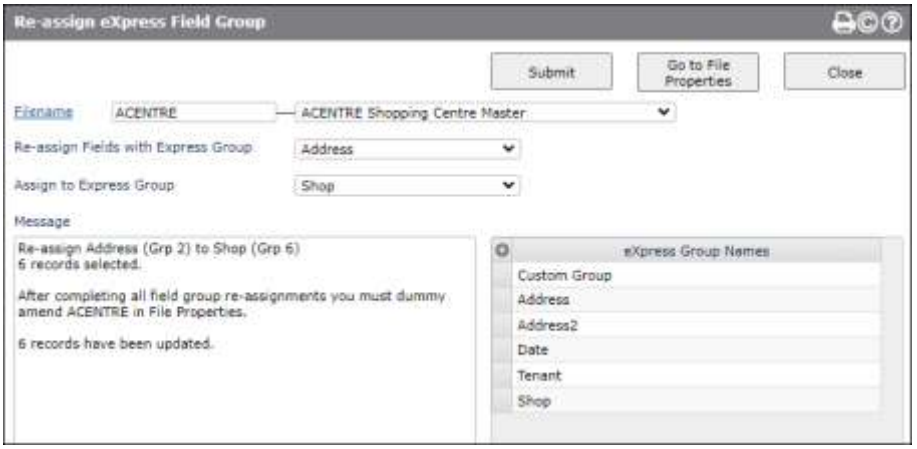
The list of available names is displayed in this example to the right. Remember that these names are associated with the position in the list. The first name in the list is assigned an integer code of 1, the second name is assigned code 2 etc. It is this integer code that is placed on the field properties record to indicate to which eXpress field group a field property belongs.

If you change the eXpress Group Name in row 1 of the list then all field properties which have been assigned to code 1 will display under the new name.

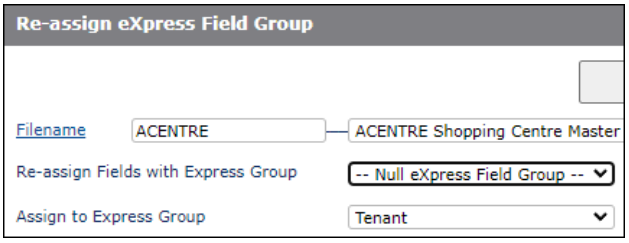
The Re-Assign Field Group function allows you to assign all fields in a selected eXpress Group to another existing eXpress Group.

In this example field properties in the *Address* group (Code 2) are re-assigned to *Shop* (Code 6).

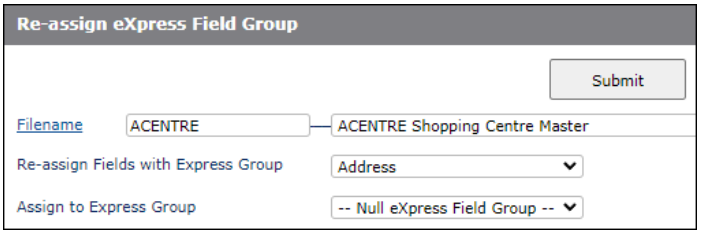
After all changes have been made use the *Go to File Properties* button to open the File Properties form where you can call up and save the file, in this example the *ACENTRE* file.



Use the *Null eXpress Field Group* option in the *Re-assign Fields with Express Group* field to select all eXpress fields not yet assigned to a Field Group, as shown to the right. In this example these fields will be assigned to the *Tenant* group.



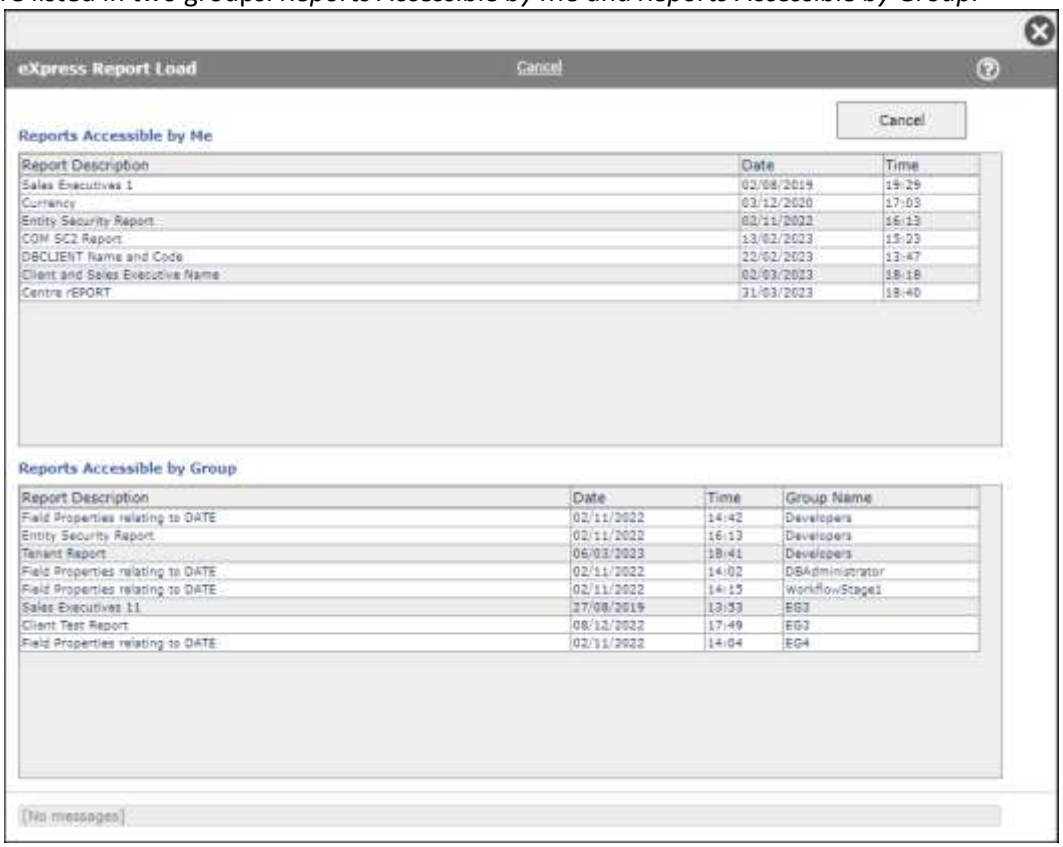
Alternatively you can remove the field group name from a set of fields by setting the *Assign to Express Group* to the *Null eXpress Field Group* option. In this example all fields assigned to *Address* are selected and the eXpress Field Group code is removed.



Run eXpress Report

Use this option to run existing eXpress reports.

These are listed in two groups. *Reports Accessible by Me* and *Reports Accessible by Group.*



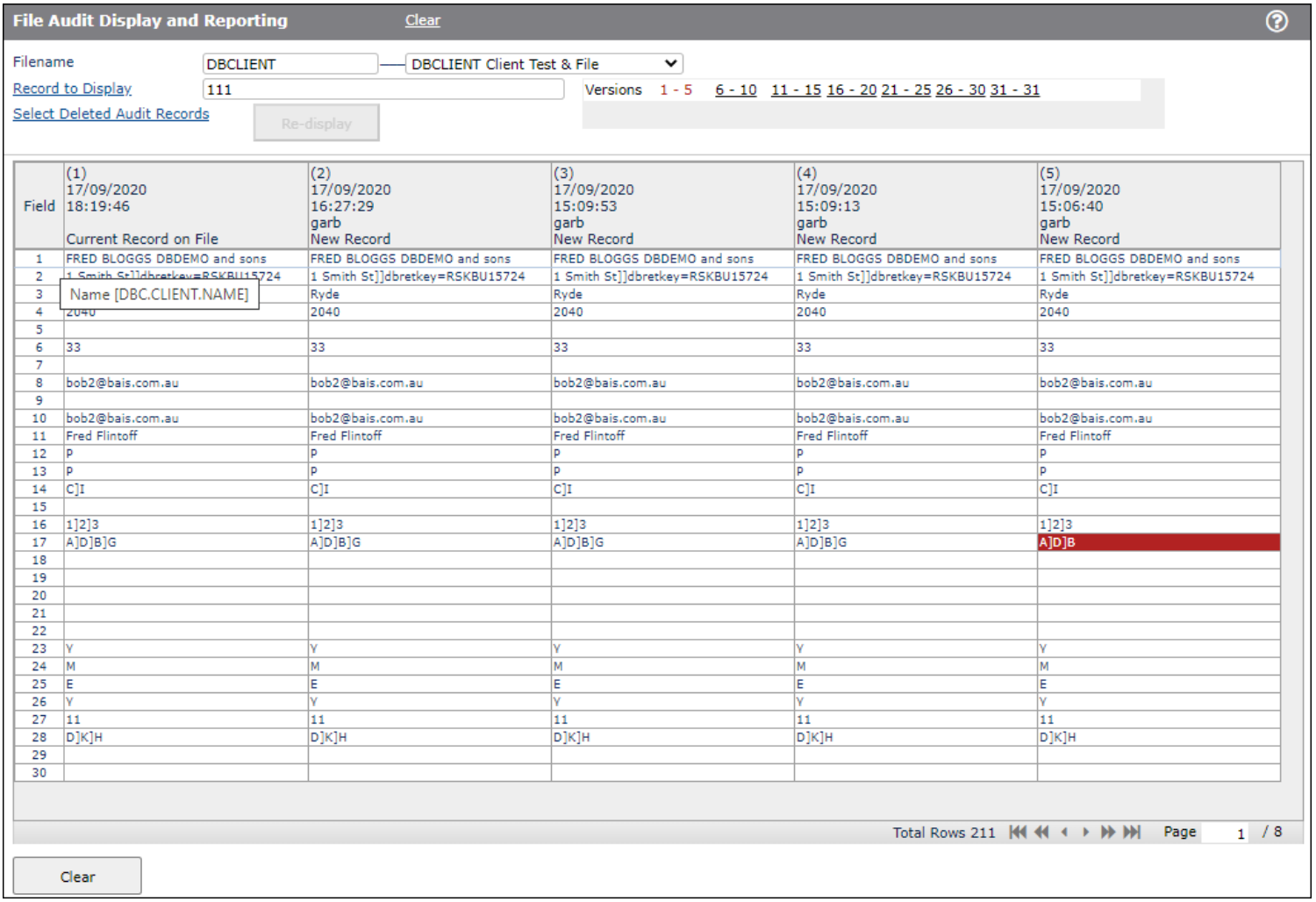
The latter group are those reports available to *User Groups* and *eXpress User Groups* of which you are a member.

Chapter 23 – Extended Auditing

Extended Auditing

When a file is flagged for extended auditing, all writes or deletes to that file, that contain modified record data, are audited.

The file DBIAUDIT.EXT is used to contain this data. Please see “File Properties” for more details on the set-up of the extended audit capabilities.



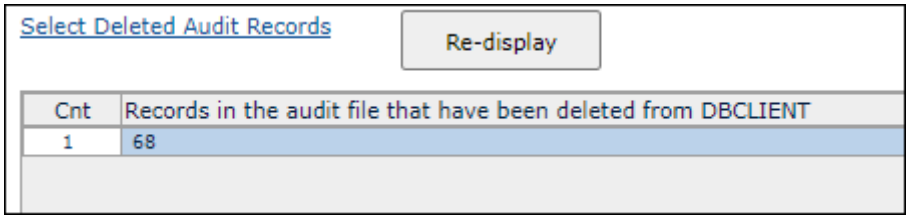
Prompts

Filename	Enter the filename required. The dropdown list provides a list of all files that are flagged for extended auditing.
In Account	Displays the account if the audit key for the selected file includes the source account.
Record to Display	Enter the identifier of the required record or click to select a record.

Select Deleted Audit Records

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

If the audit file contains records that have been deleted from the file then the above selection process will not return these. Click this hyperlink in order to display a list of records in the audit file that are no longer in the file being audited.



Click the highlighted column to select the deleted record.

Click the Re-Display button to re-display the on-form report of deleted records. This avoids performing the select again which may take some time on a large audit file.

Audit Display

Changes to fields are highlighted in red.

Hovering over the field number in the first column of the report displays the field name and description.

If the file is not updated using a DesignBais update from within a form then the update of the extended audit file must be done from the application routine that is updating the file.

The application subroutine needs to call:

```
DBI.G.AUDIT.EXTNET(PROCESS.FILENAME,ID.TO.PROCESS,WRITERECORD,EXT.TYPE)
```

where:

PROCESS.FILENAME	the name of the file being audited
ID.TO.PROCESS	the record id being updated
WRITERECORD	the record itself
EXT.TYPE	D (deleting a record) W (updating a record) N (writing a new record)

Audit Display Notes

If the DesignBais form uses a read **with no lock** then the update of the extended audit file has no original record image on DBISESSIONS on which to base its decision as to whether you are writing a changed record.

Hence it flags the audit record with “N” for *New Record* each time. Using a DesignBais optimistic lock will overcome this.

The Audit file retains an image of the record that was changed as it appeared BEFORE the change.
So the **last** audit record is not a copy of the **current** record on the file being audited.

That’s why the audit display always starts with a display of the current data – so you can see the change between the last audit record and the current state of the record. This is reflected in the heading “Current Record on File”.

When a record is deleted the audit display obviously cannot display the current record – it is gone. So it displays the state of the record at the point it was deleted.

When a new record is created the audit file is updated with a copy of the new record. Hence the current data will in this instance match the audit record. This only happens once – when the record is created and only if audit is turned on for this file at the time the record is created. (Turning audit on for an existing file full of records does not create an image of all these records in the audit file.)

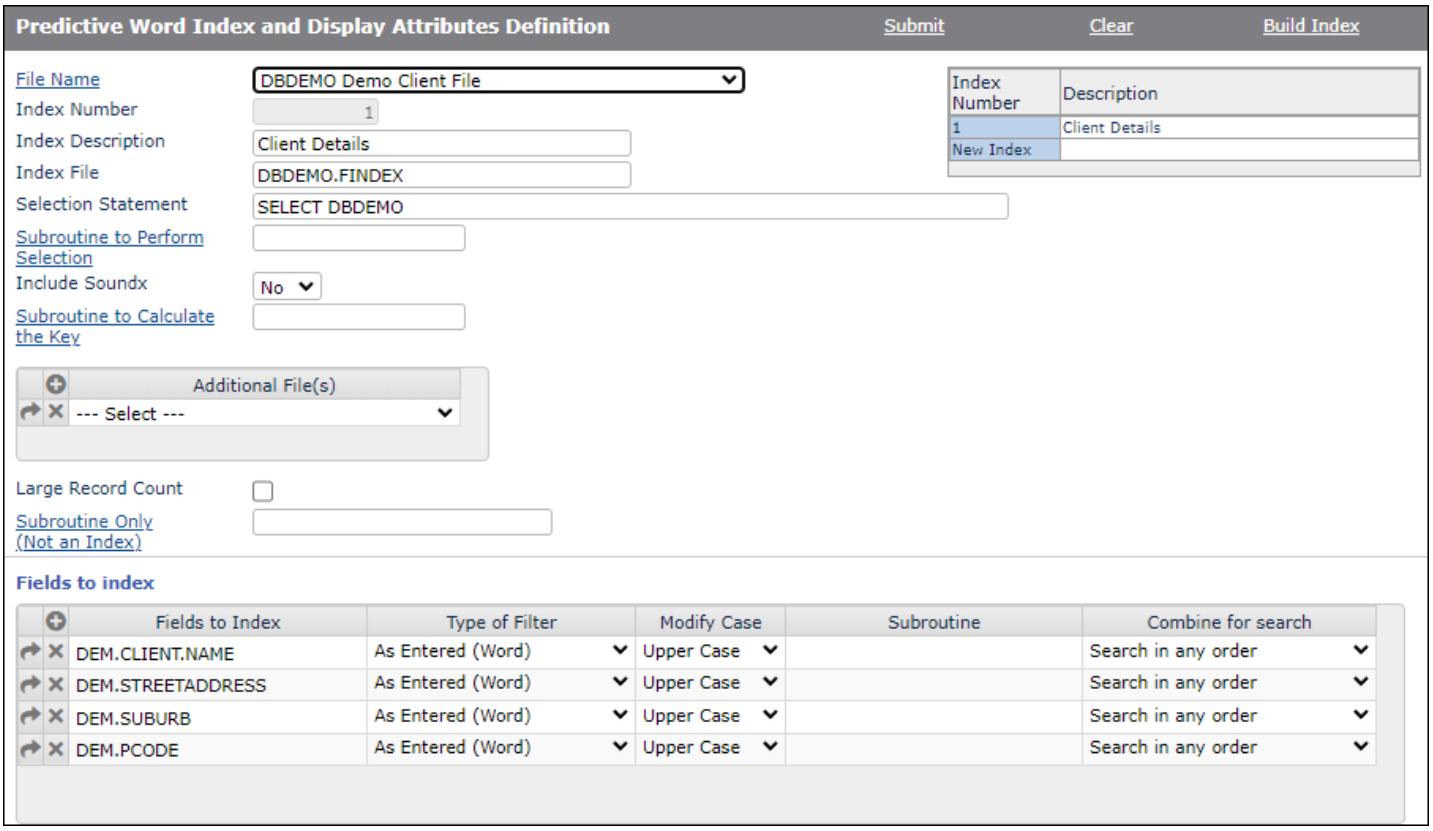
When the extended auditing status is changed for a particular file (ie turned on or off) in the File Properties form a log is updated. This records the status (on or off), the date and the user id of the user effecting the change. This log can be viewed in the File Properties form by clicking the hyperlink *Extended Audit Status Log*.

Chapter 24 – Word Index Definition

Word Index Definition / Predictive Text Searches

The Word Index Definition form is available from the Main DesignBais developer menu.

This form is split into three main parts.



File Name — Select the filename that is to be indexed. If this is a new index, click on the “New Index” link in the Index Number table.

Index Number — Is a calculated field. DesignBais will assign a New Index

Index Description — Enter the description that the user will see when the index is diplsayed.

Index File — This file will be used to store the index when built. Many indexes may be built into the same file. If the file name entered here is a new file you will need to create the file before building the index for the first time.

Selection Statement — There may be a specific Selection Statement required to select the records from the nominated file to be indexed. If not, leave this field blank. Note that if this statement is altered then the index file will have to be re-built in order to reflect the new selection statement.

Subroutine to Perform Selection — You may require a program to be invoked to perform the selection if it is not a simple access/english style sentence. If the entire file is to be selected, then do not enter a program name in this prompt.

The returned select list must be returned in DBRETURN.SELECT(1)

PROCESS.EVENT = "DBFINDEX SELECT"
PROCESS.EVENTSOURCE = Index Name. Eg. DBCLIENT*1
Return the select list in DBRETURN.SELECT(1)

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Include Soundx

If you also wish to build a Soundx equivalent to the indexed description, set this dropdown list to Yes. When the index is built or updated, the Soundx version of the index will also be built.

Subroutine to Calculate the Key

Some logic is required to determine the actual key to be indexed. If so you may call a subroutine to calculate this key.

The name of the program can be added to this field.
PROCESS.EVENT = "DBFINDEX KEY".
PROCESS.EVENTSOURCE = Name of the file
PROCESS.PARAMETER<1> = Record Identifier
DBVALUE = Value of the field
If the program sets the variable IERR.TEXT to anything other than null, the record to be indexed will be skipped.

The value returned in DBVALUE will be the value of the key.

Additional Files

You may have additional files that supply the same index data as the file being nominated. This may be in the form of archived files. You can nominate additional files to be indexed in this field.

Large Record Count

With files with large record counts, DesignBais will build the indexes in a slightly different manner. This will restrict the size of records where individual words are found in many records.

Any file with a record count of more than 50,000 records should have this check box ticked.

Subroutine Only (Not an Index)

The index definition may not be for a valid file or key structure. You may need a program to derive the keys. This can be useful when you are returning list items for standard lookups. It can be useful in identifying forms or application entries that the user has access to.

PROCESS.EVENT = "DBFINDEX LIST"
PROCESS.EVENTSOURCE = Index Name
DBVALUE<1> = List of Returned Id's
DBVALUE<2> = List of Returned Descriptions

Fields to Index

In this multivalue grid, the fields that will make up the indexed description are nominated.
In the above example, there are four fields that will contribute to the index number one on the DBCLIENT file.

Fields to Index

This field is used to define a field to be added to the index. All fields are combined to make a complete index entry, though each component can be searched separately.

Type of Filter

This describes what is to be done to the field before it is indexed. Typically you would chose Alpha or Alpha Numeric (word). This ensures that either all alpha characters or all alpha numeric characters are store. All other characters like +,-,/,$ are extracted.

Modify Case

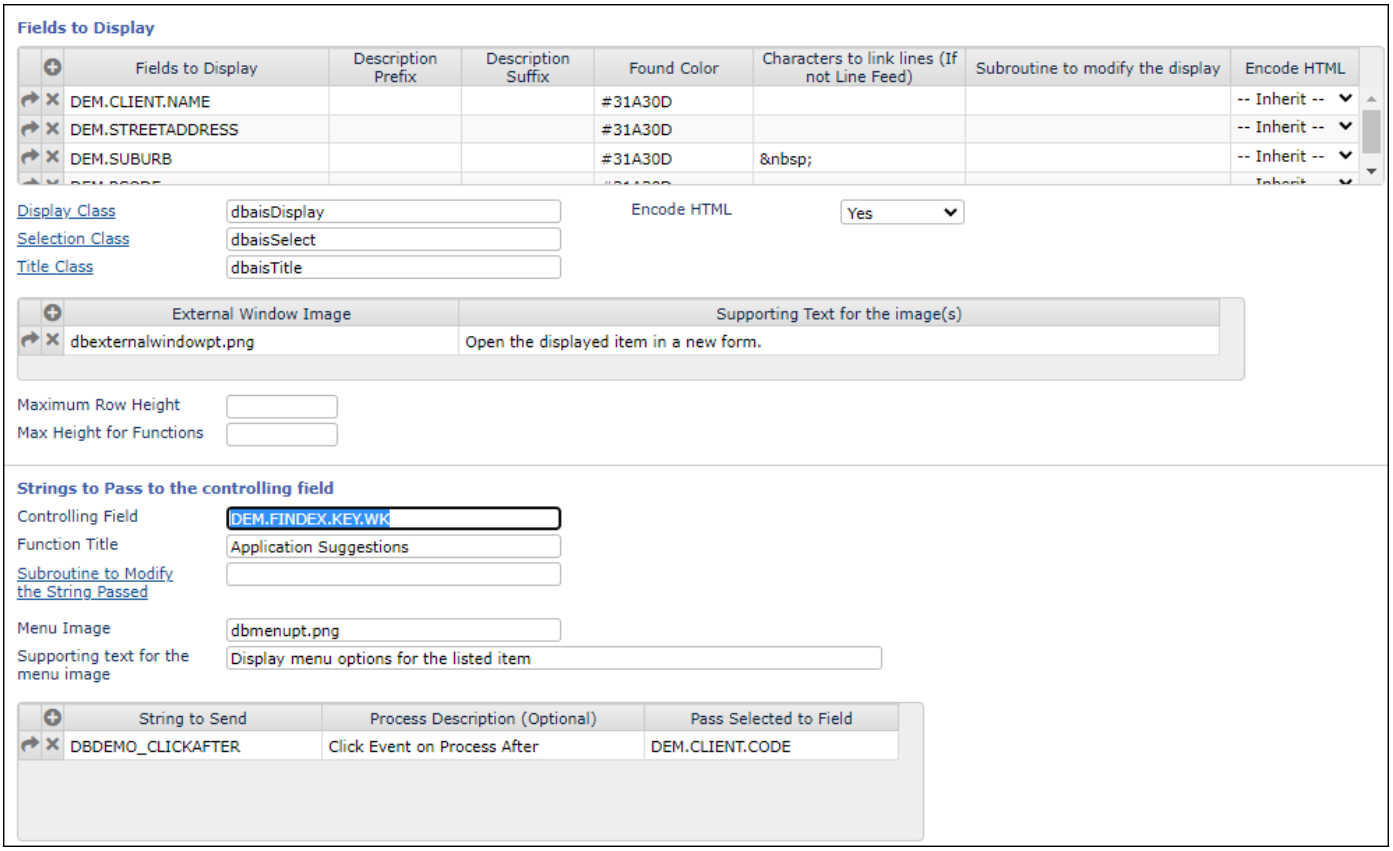
You may modify the case of the field before it is added to the index. Typically you would choose Uppercase.

Subroutine

There may be an instance where some logic is required to determine if a record can be added to an index. The name of the program can be added to this field.

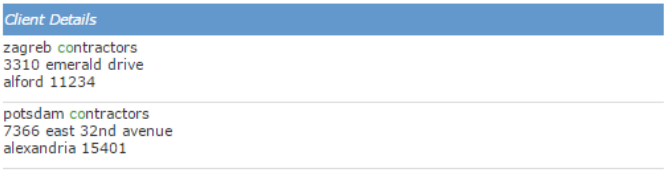
PROCESS.EVENT = "DBFINDEX ENTRY".
PROCESS.EVENTSOURCE = Name of the file * Name of Field being indexed
PROCESS.PARAMETER<1> = Record Identifier
DBVALUE = Value of the field

If the program sets the variable IERR.TEXT to anything other than null, the field to be indexed will be skipped.
The value returned in DBVALUE will be the value indexed.



In this section of the form, we describe how the results of index searches are to be displayed.

Fields to Display — The name of the fields to display in the index return.



In the above example, the fours fields are displayed over three lines.

Description Prefix — You may nominate a prefix that will appear before the fields description. This may be useful if you wish to identify phone numbers by a prefix of (ph) .

Description Suffix — As with Prefix, you may wish to add some extra detail to the field displayed to help identify the data being displayed.

Found Color — It is a very helpful visual tool for the end user if the found text is highlited. In the above example, the color #31A30D is used to display the found text. These color definitions should always be entered in uppercase.

Characters to link lines — If this field is left blank, a carriage return will be entered between one line and the next. If there is anything other than a blank cell, the character entered will be used in place of the carriage return. In the above example,   (hard space) is added between the two fields. This concatenates the suburb, a space and the postcode for the third line of the display.

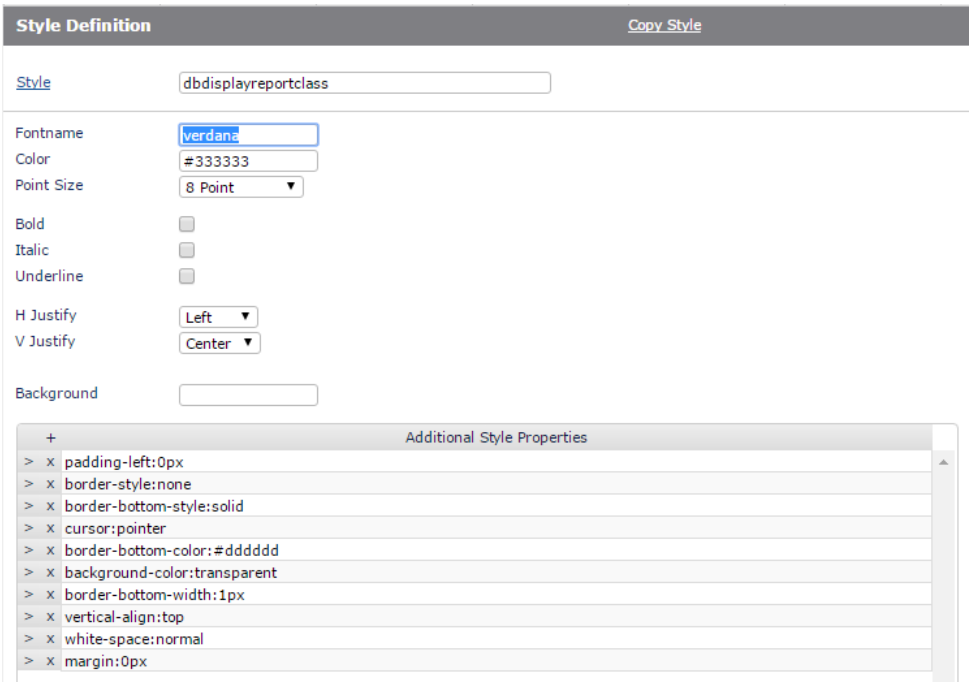
Subroutine to modify the display — The display of an indexed field may be modified before it is displayed in the routine DBI.G.DISPLAY.DBFINDEX. If so add the name of the routine in this field.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

PROCESS.EVENT = "DBFINDEX DISPLAY"
PROCESS.EVENTSOURCE = Name of the index being displayed
PROCESS.PARAMETER = Name of the field being displayed
DBVALUE = Value of the display. You may modify DBVALUE.
DBKEY = The id of the indexed item

Display Class

This is the class (Style Definition) that is to be used to display the standard list of returned values. The image below defines the Display Class used in the above example.



Selection Class

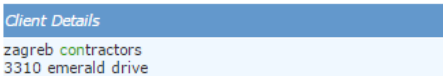
This is the class (Style Definition) that is used when the end users positions the mouse on a description in the returned list.

The example above demonstrates how this is displayed.

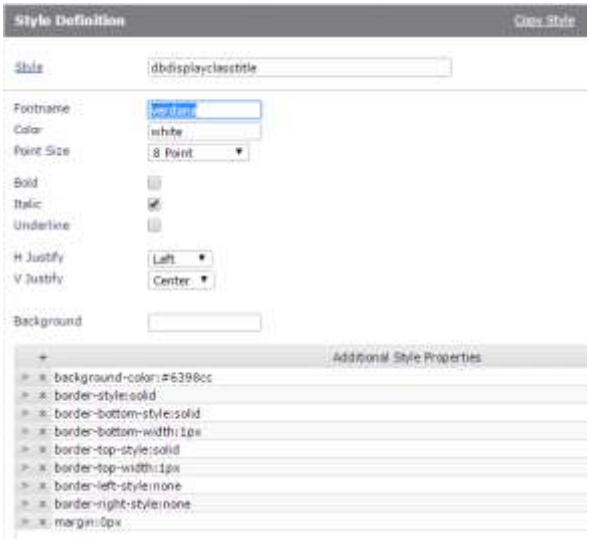
The above is the example of the style properties required to create this effect.

Title Class

This is the class (Style Definition) that is used to display the title of the list.



In the above example, the title *Client Details* is controlled by the title class.

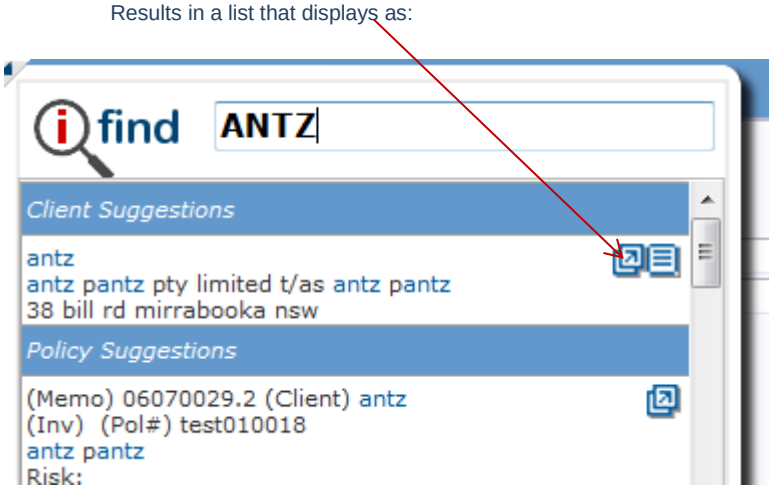


External Window Image

You may wish to include images into the display list to indicate that various selections are available. You may add image names to this definition. In the validation process called for the Controlling field, there will be an additional (3rd) parameter in DBVALUE indicating that the number of the image clicked. You can then decide how you want to process that request.

+	External Window Image	Supporting Text for the image(s)
> x	dbexternalwindowpt.png	Open the displayed item in a new form.
> x		

Results in a list that displays as:

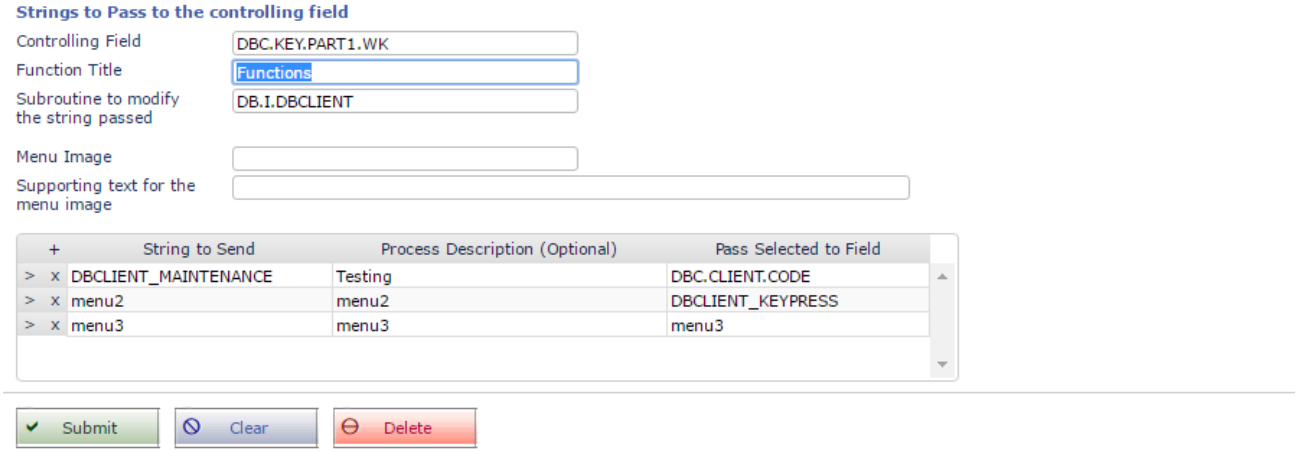


Maximum Row Height

Is used to define the maximum row height in pixels for each record in the displayed list.

Maximum Height for Functions

A selected item may return a list of functions. These functions can have a defined maximum height. If not, the height is defined by the call to DBI.G.DISPLAY.DBFINDEX (refer to the Reference Manual).



This section of the form is used to describe what happens after the user makes a selection from the returned list.

Controlling Field

On the form that has the search results displayed, there must be a nominated field to be passed the value selected from the list.
In the example above the field name is DBC.KEY.PART1.WK. This field is usually hidden on a form and is used to direct traffic based on the item selected.
This will be passed to the field as a VALIDATE event. You may trap this event in your program and decide what to do next.
If the following section of code from the website, the VALIDATE section is used to trap the response from the list.
The passed value is the in the format of Index Name | Code, Eg.DBCLIENT*1 | C00025

Here is the event that we are

Extract the Client Code from

Pass it to the client code field on the form using DBPASS.DBVALUE

Close the index display window

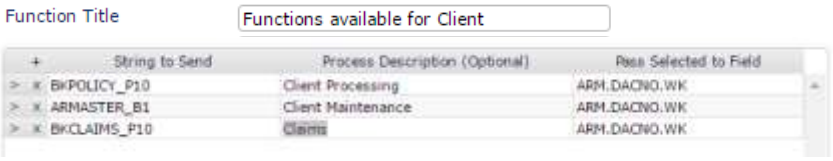
Clear the search field and place the cursor in the client code field.

```
VALIDATE:
*
      BEGIN CASE
         CASE SCREEN.NO = "TESTDRIVE"

            BEGIN CASE

               CASE EVENTSOURCE = "DBC.KEY.PART1.WK"

                  CLIENT.CODE              = FIELD(DBVALUE,"|",2)

                  DBPASS.DBVALUE           = CLIENT.CODE
                  DBPASS.DBVALUE.TO        = "DBC.CLIENT.CODE"

                  CALL DBI.G.DISPLAY.DROP("","","","","",130,79,250,150)

                  DBWORK<DBC.SEARCHKEY.WK> = ""
                  DBRETURN.TO.FIELD        = "DBC.CLIENT.CODE"

            END CASE
```

There may be a requirement for a menu of available options to be displayed when an end-user selects an entry in the list.

This is very useful if you are allowing the user a number of application functions. You may have a situation where the end user selects a client and there are a number of application functions that is possible for a user to choose from.

In the below example, when the user selects a valid client, a menu will be displayed in place of the search results. This menu will allow the user to select any one of the various options.



Subroutine to modify the string passed

You may wish to call a subroutine to modify the default strings that are passed to the menu build routine.

PROCESS.EVENT = "DBFINDEX STRING"
PROCESS.EVENTSOURCE = Index Name

DBVALUE<1> = String to Send
DBVALUE<2> = Process Description
DBVALUE<3> = Pass Selected to Field

You may modify the values in DBVALUE.

If you set IERR.TEXT to a non-null value, the click event will be ignored.

String to Send

The string that will be passed to the validate event in the controlling field. When the user makes a selection from the menu.

The string will also include the original item selected from the list.

Process Description

This is the description of the menu item.

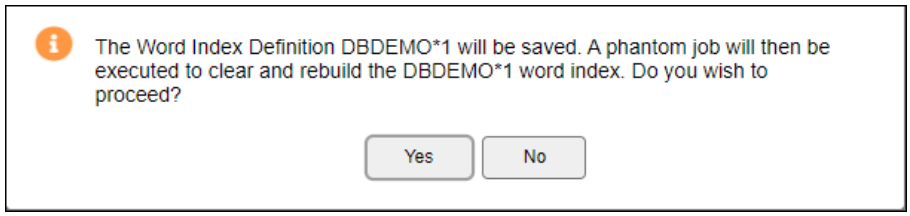
Pass Selected to Field

Allows for the specification of the field to pass the value to.

Building the Index

If you are creating a new index file then you must create the file before running the build process.

You can then use the *Build Index* button in the form header. You will be prompted:



If you respond by clicking the *Yes* button then the routine described below will be invoked after the word index definition is saved.

The routine DBI.P.DBFINDEX.BUILD is used to build the predictive index tables for indexes defined in the Word Index Definition form. This can be run from TCL. Once built, the updates will occur automatically if you are using DesignBais writes.

The index file built by this routine will have the following key structure.

If you are not using DesignBais writes then you will need to call DBI.G.DBFINDEX. This subroutine may be called from a basic subroutine on any Multi-Value database implementation.

DBI.G.DBFINDEX usage:

CALL DBI.G.DBFINDEX(IndexName, BuildType, IdTobuild)

IndexName	Any valid index. Eg DBCLIENT*1
BuildType	BUILD, UPDATE, DELETE
IdToBuild	The Id being of the record to be added or removed from the index.

If IdToBuild is null and the BuildType is BUILD, the index will be completely rebuilt.

Eg. CALL DBI.G.DBFINDEX(“DBCLIENT*1”,”UPDATE”,”C00025”)
The contents of record C00025 will be rebuilt for index DBCLIENT*1.

Index Update Routine to be run in a phantom

Word/Predictive indexes can be updated in a phantom process.

This is a three-stage process. The routine DBI.G.DBFINDEX must be invoked three times with the following parameters.

PHANTOM_START

This resets all DesignBais common variables that are used during the index update routine

Usage: CALL DBI.G.DBFINDEX(Index Name,"PHANTOM_START","")

Example: CALL DBI.G.DBFINDEX(“DBCLIENT*1”,"PHANTOM_START","")

PHANTOM_UPDATE or PHANTOM_DELETE

This will add the ‘Key to update’ to the list of keys to be indexed when the PHANTOM_COMPLETE flag is set.

Usage: CALL DBI.G.DBFINDEX(Index Name,"PHANTOM_UPDATE",Key to update)

Example: CALL DBI.G.DBFINDEX(“DBCLIENT*1”,"PHANTOM_UPDATE",”C00100”)

PHANTOM_COMPLETE

When this is invoked a phantom process will be started as all of the keys provided by the update step will be indexed.

Usage: CALL DBI.G.DBFINDEX(Index Name,"PHANTOM_COMPLETE","")

Example: CALL DBI.G.DBFINDEX("DBCLIENT","PHANTOM_COMPLETE","")

An example will clarify how to implement an index update using a phantom process.

MASTERFILE is a file with a predictive index named MASTERFILE*1.
ADD.KEY.LIST - a multivalue list of record keys to be added to the index.
DEL.KEY.LIST - a multivalue list of record keys to be removed from the index.

```
INDEX.NAME = ‘MASTERFILE*1’
CALL DBI.G.DBFINDEX(INDEX.NAME,’PHANTOM_START’,’’)
AMAX = DCOUNT(ADD.KEY.LIST,VM)
FOR DL = 1 TO AMAX
     THIS.KEY = ADD.KEY.LIST<1,DL>
     CALL DBI.G.DBFINDEX(INDEX.NAME,"PHANTOM_UPDATE",THIS.KEY)
NEXT DL
*
DMAX = DCOUNT(DEL.KEY.LIST,VM)
FOR DL = 1 TO DMAX
     THIS.KEY = DEL.KEY.LIST<1,DL>
     CALL DBI.G.DBFINDEX(INDEX.NAME,"PHANTOM_DELETE",THIS.KEY)
NEXT DL
*
CALL DBI.G.DBFINDEX(INDEX.NAME,"PHANTOM_COMPLETE","")
```

The update of the index is triggered by the PHANTOM_COMPLETE parameter.

Calling the Index Lookup routine

Now that we have defined and built the predictive index definitions, we need to direct user entry from a field towards the DesignBais routines used to extract and display the search results.

DBI.G.GET.DBFINDEX

The routine DBI.G.GET.DBFINDEX is used to extract a list of values that match a passed value for nominated predictive/word based indexes.

Usage: CALL DBI.G.GET.DBFINDEX(Index Name(s) ,Passed Value,Returned Keys,Search Depth,Passed From)

Index Name(s)

Name of Indexes for the search. Eg. DBCLIENT*1 : VM : DBSUPPLIER*2

Passed Value

The value that has been entered by the end user. If you are using the KEYPRESS event then this value is DBVALUE. It is recommended that you use an assignment like PASSVALUE = DBVALUE and use PASSVALUE in the call. This will avoid any issues if DBVALUE is changed by any of the search routines.

Returned Keys

This variable contains a list of keys returned by the search. The keys will be prefixed by the index name, Eg. DBCLIENT*1|COOO23 : AM : DBCLIENT*1|C00024

The keys provided will have already passed the DesignBais security routines. The main reason to split the key extraction and display routines is so the programmer can intercept the key list and apply any application security rules.

Search Depth

This variable controls how many records are returned by the search. As this is a predictive index search, you need to ensure that the response to the user is immediate. It is recommended that this number should not be more than about 20. If you are searching multiple indexes, this number should be no more than about 10. The idea of predictive indexing is that the user continues to type which will refine the results (similar to search engines). If you return too many items, the search becomes too slow and defeats the purpose of using this type of indexing mechanism.

Passed From

This field tells the search routine where the passed value was sourced from.

This value can be left null, which means that the search routine will search all indexed fields in an index for the value entered. Alternatively, you can direct the search by specifying that the passed value is from a specific field. This is useful if you are looking for a street name and are providing the suburb. You would then indicate the name of the field containing the suburb in the Passed From field. This will then ensure that all streets returned belong to the suburb provided.

Here is an example of the code to create the Client Details shown in the above examples:

```
85  KEYPRESS:
86  *
87        BEGIN CASE
88           CASE (SCREEN.NO = "PREDICTIVE" OR SCREEN.NO = "TIMERTEST") AND EVENTSOURCE = "DBC.SEARCH.WK"
89              IF LEN(DBVALUE)=0 THEN
90                 CALL DBI.G.DISPLAY.DROP("","","","","",1,1,1,1)
91                 RETURN
92              END
93              PASSVALUE=DBVALUE
94              INDEX.NAME='DBCLIENT*1'
95              CALL DBI.G.GET.DBFINDEX(INDEX.NAME,PASSVALUE,KEYLIST,20,'')
96              *
97              INDEX.ID = "DBCLIENT*1"
98              READ DISPLAYPARAMS FROM F.DBIPARMS,INDEX.ID THEN
99                 DISPLAYPARAMS = CHANGE(DISPLAYPARAMS,AM,VM)
100             END ELSE
101                DISPLAYPARAMS     =530;* col
102                DISPLAYPARAMS<1,2>=80 ;* row
103                DISPLAYPARAMS<1,3>=484;* width=colspan
104                DISPLAYPARAMS<1,4>=550;* depth=rowspan
105             END
106             *
107             CALL DBI.G.DISPLAY.DBFINDEX(INDEX.NAME,PASSVALUE,KEYLIST,RETURNARRAY,DISPLAYPARAMS,"")
108
109       END CASE
110       RETURN
```

Another example is taken from the former DesignBais website. This is the code, in the KEYPRESS event, to get the search results:

```
KEYPRESS:

      BEGIN CASE

         CASE SCREEN.NO = "TESTDRIVE" AND EVENTSOURCE = "DBC.SEARCHKEY.WK"
            IF LEN(DBVALUE) = 0 THEN
               CALL DBI.G.DISPLAY.DROP("","","","","",130,79,250,150)
               RETURN
            END
            index.lookup = "DBCLIENT*1":VM:"DBCREDITOR*6":VM:"DBIFORMS*100"

            PASSVALUE = DBVALUE
            CALL DBI.G.GET.DBFINDEX("DBCLIENT*1",PASSVALUE,KEYLIST,20,"")
*
```

Call the get index routine to return the list of keys that match the searched string.

Now that we have a list of valid option keys returned from the search, we can call a routine to display the results.

The routine DBI.G.DISPLAY.DBFINDEX is used to display a list of keys provided by DBI.G.GET.DBFINDEX.

Usage:

CALL DBI.G.DISPLAY.DBFINDEX(Index Name(s) ,Passed Value ,Key List, Return Array ,Display Parameters,Passed From)

Index Names As per DBI.G.GET.DBFINDEX

Passed Value As per DBI.G.GET.DBFINDEX

Key List Is the list of keys provided by DBI.G.GET.DBFINDEX or your own routine.

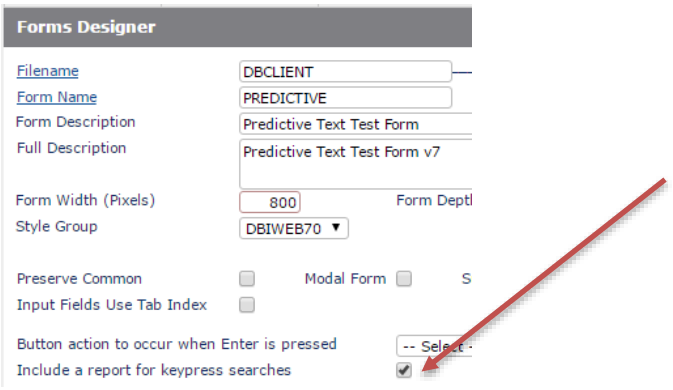
Return Array An attributed list of displayed items

Display Parameters

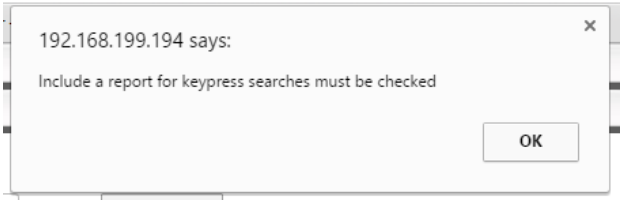
Attr <1> = Column to display the search results (Pixel position X axis)
<2> = Row to display the search results (Pixel Position Y axis)
<3> = Width of display box (pixels)
<4> = Depth of display box (pixels)

Passed From As per DBI.G.GET.DBFINDEX

You must have the '**Include a report for keypress searches**' checked on the Forms Designer front screen for this function to work since it uses the 11th On-form Report subscript, which is reserved for DesignBais usage.



If this check box is not checked then you will see a warning alert when you attempt to access the field which is indexed.



Below is the code that is used to display the index results in the website.

```
            DISPLAYPARAMETERS      = 525
            DISPLAYPARAMETERS<1,2> = 200
            DISPLAYPARAMETERS<1,3> = 264
            DISPLAYPARAMETERS<1,4> = 210
*
            CALL DBI.G.DISPLAY.DBFINDEX("DBCLIENT*1",PASSVALUE,KEYLIST,RETURNARRAY,DISPLAYPARAMETERS,"")
```

There may be instances where you wish to close the predictive text display box automatically.
Use DBI.G.DISPLAY.DROP to do this.

Usage: CALL DBI.G.DISPLAY.DROP("","","","","",1,1,1,1)

The parameters are reserved. The routine should always be called as displayed above.

In the website it is used when any of the update buttons are pressed. It is also used when the Search field is nulled by the user.

Implementing Predictive Text without Word Index

Refer to the Demo Form DBDEMO*WORDINDEXTEST as a guide to implementing predictive text lookup without creating a word index definition.

Your form must have the *Include a report for keypress searches* field checked.
You must have a hidden field on the form that is defined as the field to which the selected value is returned and made available to the database.
Your subroutine must establish the position of the dropdown selection pane and clear this pane:

```
* set the position of the dropdown selection pane
*
DISPLAYPARAMS     = 10;* col
DISPLAYPARAMS<1,2>= 80;* row
DISPLAYPARAMS<1,3>=340;* width=colspan
DISPLAYPARAMS<1,4>=465;* depth=rowspan
*
* clear the selection pane
*
IF LEN(DBVALUE) < 1 THEN
   CALL DBI.G.DISPLAY.DROP("","","","","",DISPLAYPARAMS<1,1>,DISPLAYPARAMS<1,2>,DISPLAYPARAMS<1,3>,DISPLAYPARAMS<1,4>)
   RETURN
END
*
* set the minimum number of characters before a list is displayed
*
IF LEN(DBVALUE) < 2 THEN RETURN
```

Note that you should set a minimum length for the search text string before you attempt to select and display any results. This ensures that your form remains efficient by not attempting to display an unreasonably large number of search results. In the code example above the display only renders results when the search string is 3 characters or greater in length.

Your subroutine will require logic to select records based on the search string entered by the user.
In the example below a simple *INDEX* command is used to check for the presence of the entered search text in records read from the database file. The selected records are stored in an array so that no further reads are required after the user selects the required record.

Pipe characters are used by DesignBais to extract the key of the selected record. Therefore you must not use the pipe character (|) in your keys that are passed in the KEY.DETAILS argument. For each column that you wish to be responsive to the user click in the results pane, pass the name of the hidden field followed by a pipe character followed by the key of the record that corresponds to this row of the display.

Call the subroutine DBI.G.DISPLAY.DROP to render an on-form report that displays the details of the strings returned from the code executed in your keypress event.

```
* establish an array so that records need only be read once
*
EQU DISP.LIMIT TO 40
DIM RES.ARRAY(DISP.LIMIT)
MAT RES.ARRAY = ''
*
TESTSTR = OCONV(DBVALUE,'MCU')
*
RESULT.LIST = ""
*
RMAX = 0
*
* select the file - may need database indexing to maintain response time
*
CMD = 'SSELECT DBDEMO'
EXECUTE CMD CAPTURING THIS RETURNING THAT
LOOP
```

```
         READNEXT ID ELSE EXIT
         READ REC FROM F.DBDEMO,ID ELSE CONTINUE
         DROPREC = ID
         DROPREC<1,-1> = REC<DEM.CLIENT.NAME>
         DROPREC<1,-1> = REC<DEM.SUBURB>
         DROPREC<1,-1> = REC<DEM.PCODE>
         TEST.DROPREC = OCONV(DROPREC,'MCU')
         IF INDEX(TEST.DROPREC,TESTSTR,1) THEN
            *
            * the record, or selected fields, contain the search string so include in display
            *
            RESULT.LIST<1,-1> = ID
            RMAX += 1
            RES.ARRAY(RMAX) = DROPREC
         END
         IF RMAX = DISP.LIMIT THEN EXIT
      REPEAT
      *
      * populate the dropdown display pane
      *
      HEADER.DETAILS = "Code":VM:"Name":VM:"Suburb":VM:"Pcode"
      DISPLAY.DETAILS = ""
      KEY.DETAILS = ""
      JUST.DETAILS = ""
      WIDTH.DETAILS = "10":VM:"30":VM:"20":VM:"10"
      *
      FOR DL = 1 TO RMAX
         THIS.RESULT = RESULT.LIST<1,DL>
         OUTPUT.LINE = THIS.RESULT
         OUTPUT.LINE<1,2> = RES.ARRAY(DL)<1,2>
         OUTPUT.LINE<1,3> = RES.ARRAY(DL)<1,3>
         OUTPUT.LINE<1,4> = RES.ARRAY(DL)<1,4>
         DISPLAY.DETAILS<DL> = OUTPUT.LINE
         *
         * key.details must define the name of the field on your form to which the key value will be passed, pipe, then the value of the
key
         *
         KEY.DETAILS<DL> = STR("DEM.FINDEX.KEY.WK|":THIS.RESULT:VM,4)
      NEXT DL
      *
      CALL
DBI.G.DISPLAY.DROP(HEADER.DETAILS,DISPLAY.DETAILS,KEY.DETAILS,WIDTH.DETAILS,JUST.DETAILS,DISPLAYPARAMS<1,1>,DISPLAYPARAMS<1,2>,DISPLAYPARA
MS<1,3>,DISPLAYPARAMS<1,4>)
      *
      RETURN
```

The DBI.G.DISPLAY.DROP subroutine uses the special on-form report *rdesignbaisdrop* which is held in the 11th occurrence of the on-form report arrays which is reserved for DesignBais use. When the user clicks on the desired row in the display the DesignBais engine passes the key back in the keypress field with *PROCESS.EVENT* set to *VALIDATE*. This behaviour differs from the normal processing of on-form reports where the click event is handled in *REPORT* event.

Chapter 25 – DBIGLOBAL File

DBIGLOBAL File

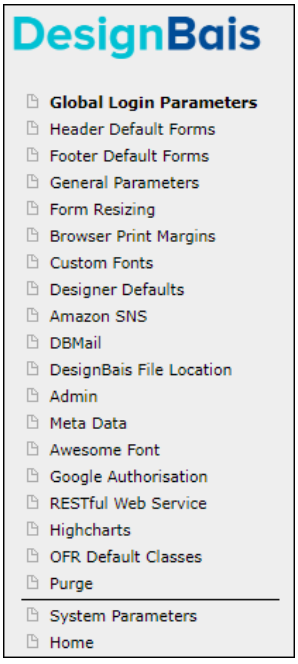
This file is used to define default parameters for the entire system. In releases prior to release 6, parameters were stored only in DBIPARMS. The issue with this was that there may be a requirement on SAS servers to have variations in the system parameters. Another more global parameter file was required to allow for global date formats and insert strings.

This file is used for variables that are global within your system. This file should be created once in an account that contains global definitions within your application environment. If you do not have such an account, the DBILOGIN account could be used for this purpose.

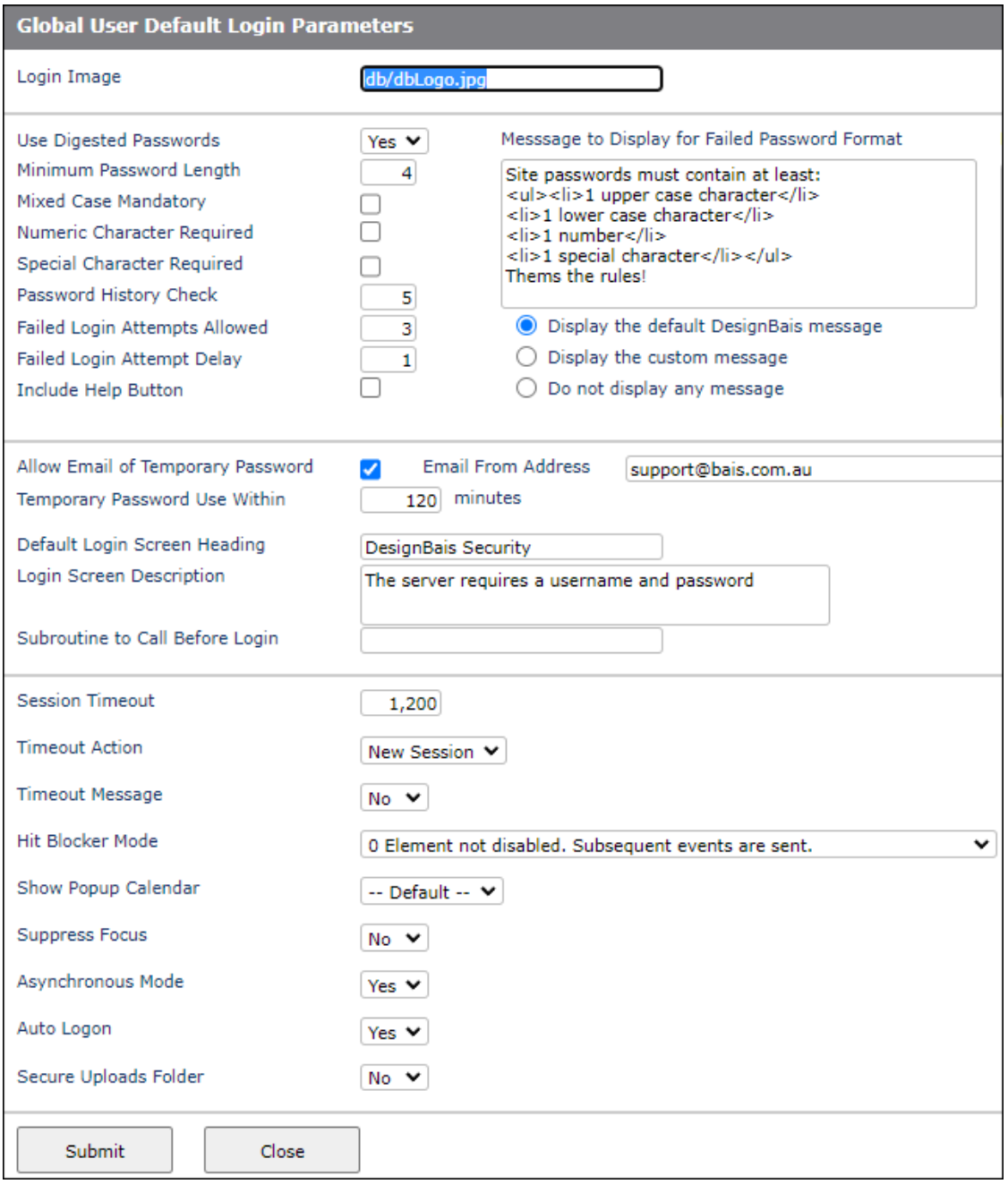
This file does not have to exist for DesignBais to operate normally.

If the file DBIGLOBAL is accessible from the currently logged in account (for a DesignBais user) it will be referred to in the following instances.

The Global Parameters menu options are displayed in the Development Tools side menu.

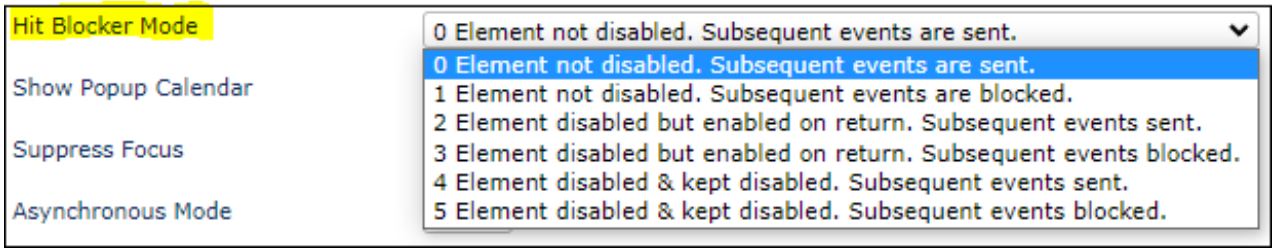


Global Login Parameters



Login Image	The image to use to display on the user login screen when DesignBais user login is activated.
Use Digested Passwords	Set to *Yes* to invoke user passwords hashed using SHA-1 algorithm. If not set then passwords are encrypted. Changing this flag from *No* to *Yes* propts the user to run the DesignBais Upgrade Routine 36. Refer to the Migration Manual.
Minimum Password Length	The minimum length permitted for a DesignBais user password.
Mixed Case Mandatory	If the check box is checked then the DesignBais user's password will be forced to contain at least 1 upper case and at least 1 lower case character.

Numeric Character Required	If the check box is checked then the DesignBais user's password must contain at least 1 numeric character.
Special Character Required	If checked the DesignBais user's password must contain a special character (i.e. not 0-9 or A-Z).
Password History Check	The number of previous passwords to check to ensure a previous password is not re-used
Failed Login Attempts Allowed	The number of failed login attempts allowed before the login process is aborted. If null then the system defaults to 5. As the number of consecutive failed attempts increases, the time delay before another attempt can be made, increases.
Failed Login Attempt Delay	The delay in seconds after a failed login, say a wrong password is entered, before another attempt is allowed.
Include Help Button	Check this box to force the display of the Help button (?) on the DesignBais login form.
Message to Display for Failed Password Format	Select to either display the default DesignBais message, display the custom message that can be entered in the text field above the checkbox, or display no message at all.
Password Format Message	You may enter a message here that will display when a user fails to create a password that obeys the password criteria of length, case and special characters.
Failed Login Attempt Delay Control	This radio button controls the message displayed when a password does not match the required specifiactions.
Allow Email of Temporary Password	If checked then the users password may be emailed to the email address on the user record if it has been lost or forgotten.
Email From Address	The email address that will be used as the sender for forgotten passwords.
Default Login Screen Heading	The default heading to display on the DesignBais user login screen.
Login Screen Description	The default description to appear on the DesignBais user login screen.
Subroutine to Call Before Login	The basic subroutine to call after a user's login password has been successfully validated. PROCESS.PARAMETER will contain the user's login id. A value in IERR.TEXT will stop login.
Session Timeout	A session will expire after an idle period of the session timeout in minutes. Valid values are 0 to 1200 minutes (20 hours). The recommended value is 20 minutes or longer. A value entered in the System Parameters will take precedence over a value entered here in the Global Login Parameters. A default value of 20 minutes will be used if no Session Timeout has been supplied. See **DesignBais Page Refresh** below which provides more detail.
Timeout Action	This setting determines the action to take when a DesignBais browser tab session has been idle for the Session Timeout period. *New Session* will display a timeout message if the user attempts to activate the session. The timeout warning message may be suppressed. The user will be automatically taken to their start form. *Logout* will take the user to the logout page. *Error Page* will take the user to the DesignBais error page. The value set here in Global Login Parameters can be inherited in the System Parameters settings.
Timeout Message	When a DesignBais browser tab session has been idle for the Session Timeout period then a timeout message will display if the user attempts to activate the session. The timeout warning message may be suppressed by selecting No. The user will be automatically taken to their start form. May be overwritten for individual users.
Hit Blocker Mode	This feature enables the developer to control the action of subsequent events following a button event, input event, and a range of other events such as Image or Report. Hit Blocker Mode. May also be set on System Parameters or applied at Field Level. The Global setting will be applied these other levels are set to 'Inherit'. 'Inherit' at the Field level will inherit the System Parameters setting. If this is 'Inherit' then the Global Parameters setting is used.



Values are:
0 = Element not disabled. Subsequent events are queued and sent in correct order.
1 = Element not disabled. Subsequent events are blocked. Events are lost and page is disabled.
2 = Element disabled but enabled on return. Subsequent events queued and sent in correct order.
3 = Element disabled but enabled on return. Subsequent events are blocked.
4 = Element disabled & kept disabled. Subsequent events queued and sent in correct order.
5 = Element disabled and kept disabled on return. Subsequent events are blocked.

Show Popup Calendar

This feature enables the display of the Date Picker Calendar. The default behaviour is to not display the calendar. Selecting On Focus causes the Date Picker to display on any date field as it gains focus. Note that when the default setting is used the Date Picker can still be displayed using a button. Refer to the subroutine **DBI.G.CALENDAR** in this Manual.

Suppress Focus

DesignBais by default controls field focus by sending a javascript focus() command after a change event triggered when the user tabs out of a field or presses the enter key in a field. This requires strict control of the sequencing of events between the web component and the database component which may interfere with users typing ahead.

Selecting Yes will suppress the sending of focus() commands, while No is the default behaviour. This setting may be over ridden by the System Parameters value.

The use of DBRETURN.TO.FIELD in basic code will take precedence.

Asynchronous Mode

By default DesignBais runs with asyncronous ajax calls since synchronous mode has been deprecated by browser vendors. Developers may, however, choose to run in synchronous mode by selecting *No* in this field. This setting in the Global Login Parameters may be inherited by the equivalent field in System Parameters. Hit Blocker modes will be ignored if synchronous mode have been selected.

Note that a DesignBais application that relies on synchronous ajax calls may cease to function correctly once the support for synchronous ajax calls is no longer available in the web browser being used.

Auto Logon

When starting a second or subsequent tab session on a browser the logon form can be by-passed and DesignBais will log on the same user as was logged on in the initial tab. This flag can be inherited in the the equivalent setting in System Parameters.

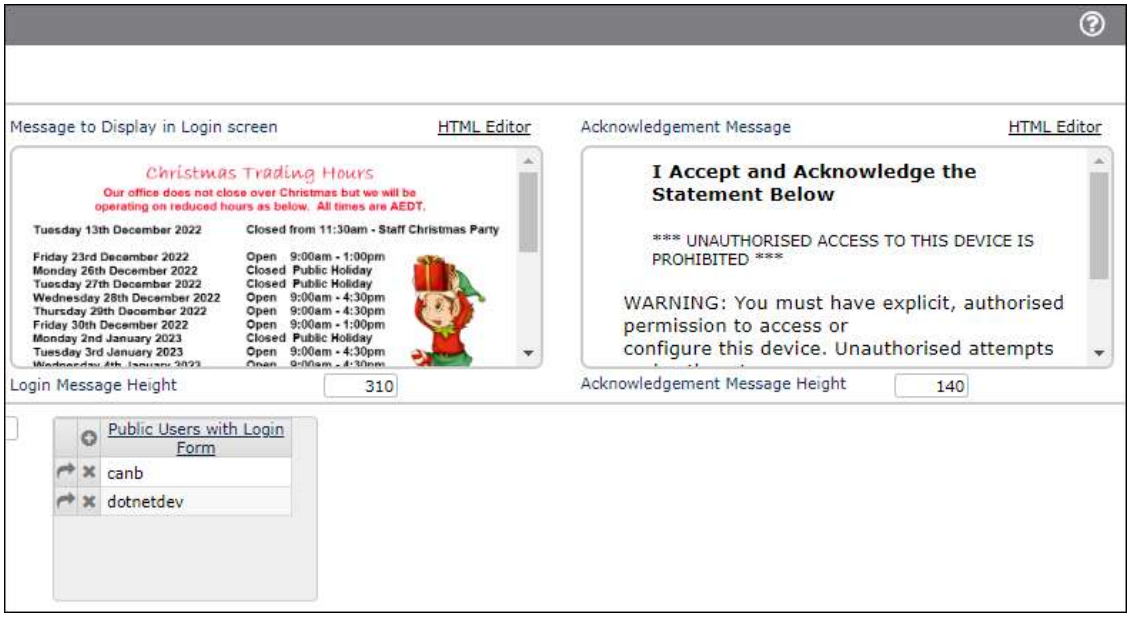
Auto logon requires that the files DBIPARMS, DBIGLOBAL, DBISTATS and DBISESSIONS are shared within the accounts that users are accessing, and that the DBIUSERS and DBIGROUPS files are either shared or that user and group ids exist in all instances of these 2 files.

In addition the public user id, defined in the db.config file as the connection user, must be listed in the *Public Users with Login Form* field in System Parameters and/or in Global Login Parameters.

To assist in reviewing these requirements for a particular user id refer to the *User Maintenance* form where there is a clickable hyperlink button *Display Start Account List*. Refer to the *User Maintenance* section of this manual.

Secure Uploads Folder

If checked (set to Y) then DesignBais will prevent unauthenticated access to the website *uploads* folder. Users will need to log in to the application before they upload or download files. If unchecked than this feature is not active. This setting may be over written in System Parameters.



Message to Display in Login Screen

You can use the HTML Editor to create a message to display in the DesignBais login form DBIGLOBAL_D21.

Login Message Height

The height needed to display the Login Message. This allows the form to be adjusted for any HTML message. It is recommended to keep the height to a reasonable limit so that the login form still fits on the page.

Acknowledgement Message

If an acknowledement message exists it will display above the DesignBais login form and will require the login user to check a box to acknowledge the message before the login process can proceed. The only alternative is to close the browser tab.

Acknowledgement Message Height

The height needed to display the Acknowledgement Message. This allows the form to be adjusted for any HTML message. It is recommended to keep the height to a reasonable limit so that the login form still fits on the page.

Public Users with Login Form

The list of Public Login users that start in a login form where the operator switches to their own application user ID. This list is used by the Auto Login feature to automatically switch to the currently connected user when a new tab is opened in a browser.

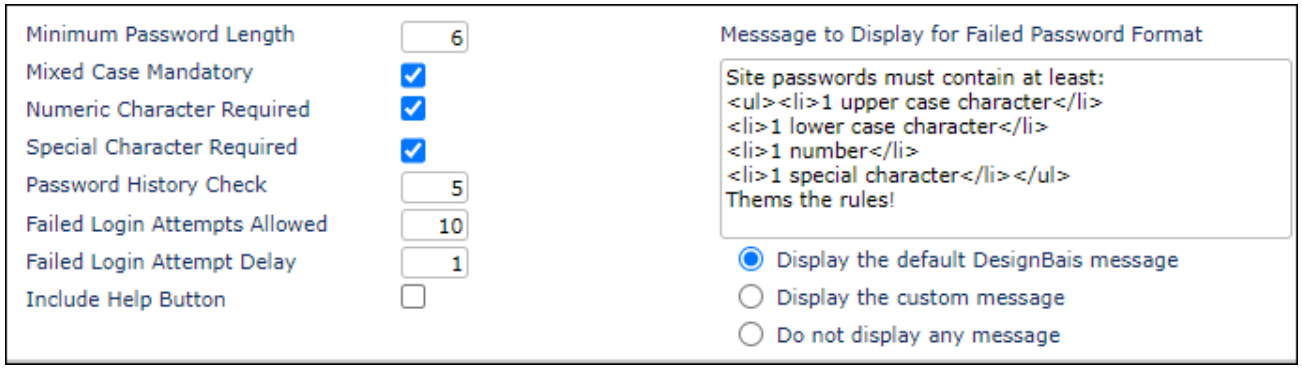
DesignBais User Login Form

There is a form DBIGLOBAL_D21 which can be used as a user login screen. If the user enters the correct credentials then control is passed to the user's start form. Alternatively the developer can supply a Subroutine to Call Before Login (see above). This basic subroutine is called after a user's login password has been successfully validated. A value in IERR.TEXT will stop login.

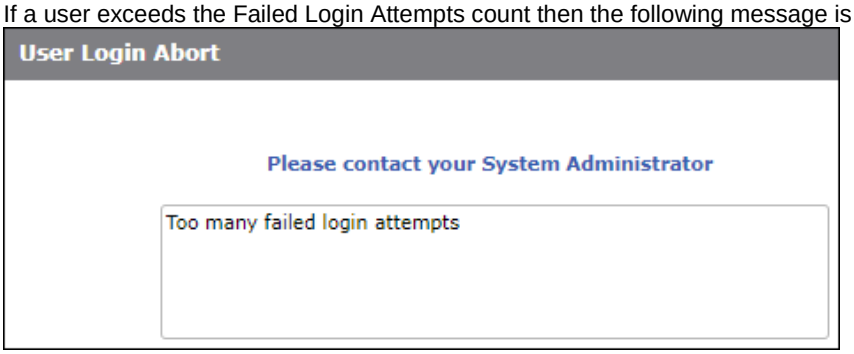
The Login Form enables DesignBais to change the logged in user from the connection user id to the user id entered into the login form. This is done using DBALTUSER. Setting DBALTUSER will change the user id and reset WEBLOGON. Note that the login account specified by the db.config for the specified entry point must share the DBISESSIONS file with the start account of the user id entered into the login form.

The login form provides a method for obtaining the user's credentials in order to open their designated start form. This assumes that the login account and the user's start account share the same DBIGLOBAL file. This would normally be the case, but if not then it could have an effect on, for example, whether a user is prompted for Google Two Factor authentication.

Password format is controlled by parameters that are maintained in the Global Login Parameters form.



If a user exceeds the Failed Login Attempts count then the following message is displayed:



In the example below the start account of the user id entered into the login form must have the same DBISESSIONS file as the account BA.LOGINNET.

```
<entryPoint qcode="">
   <loginHost>192.168.20.10</loginHost>
   <BASUBROUTINE>BAWEBEXECNET</BASUBROUTINE>
   <loginHostType>UNIVERSE</loginHostType>
   <loginAccount>BA.LOGINNET</loginAccount>
   <loginUser>DesignBais</loginUser>
   <loginPassword>password</loginPassword>
   <loginPublicUser>dbnetuser</loginPublicUser>
```

Code similar to that shown below is required to change the WEBLOGON user id:

```
   IF ACCOUNT.SELECTED # '' THEN
      WEBLOGON=DBWORK<DBIGO.USER.WK>
      * save variables
      SAVE.SCREENROOT=SCREENROOT
      SAVE.EVENT=PROCESS.EVENT
      SAVE.EVENTSOURCE=PROCESS.EVENTSOURCE
      SAVE.WEBLOGON=WEBLOGON
      * invoke the Account Selection event for the new user
      SCREENROOT='DBIUSERS_D20'
      PROCESS.EVENT='BUTTON'
      PROCESS.EVENTSOURCE='B.LOGTO'
      DBWORK<DBIU.ACCOUNTS.WK> = ACCOUNT.SELECTED
      * set the new user
      WEBLOGON=DBWORK<DBIGO.USER.WK>
      CALL DBI.I.DBIUSERS
      * restore saved variables
      SCREENROOT=SAVE.SCREENROOT
      PROCESS.EVENT=SAVE.EVENT
      PROCESS.EVENTSOURCE=SAVE.EVENTSOURCE
      WEBLOGON=SAVE.WEBLOGON
      * flag to DesignBais that the the logged in user is to change
      DBALTUSER=DBWORK<DBIGO.USER.WK>
      * clear variables
      DBRECORD=''
      DBOTHER.RECORD(2)=''
   END
```

Developers should be aware of the following shortcomings in the D20 login form:

- DBIGLOBAL_D20 includes buttons “Change Password” and “Forgotten Password”. The password is encrypted and stored in DesignBais.

- When a user enters a user id and clicks “Forgotten Password” then the password is un-encrypted and sent to the user’s email address. The email states, for example, where the user password is “freddy”:

    Your DesignBais password is freddy

The use of the DBIGLOBAL_D20 Login form is therefore not recommended. Developers should implement DBIGLOBAL_D21 and digested passwords.

If developers have security concerns they can create their own log in form, incorporating the code shown above that sets the DBALTUSER variable, and add security features, as required.

Password Security

From Release 8.9.1.1, for added security, passwords have been removed from DBIXMLLOG records and are not passed to the custom logging routine. This means that password fields are no longer returned to the web unless initialising. This means that passwords may be cleared but otherwise are not sent to the web.

The DesignBais DBIGLOGAL login form D21, only sends the password for verification – the database does not send the password to the web component. The incoming password field no longer appears in the DBIXMLLOG file nor in the debug.txt.

The DesignBais login form has a process to email a random password to a user if they want to change or have forgotten their password. This temporary password has a limited usage period.

There is also an option to store the password as a digested string. This prevents database developers from accessing passwords by editing records.

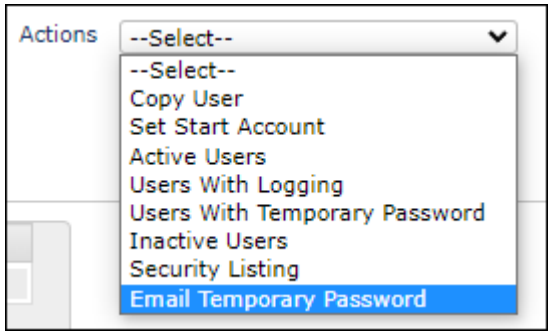
These measures ensure that a user’s password is only exposed during log on when they enter it in the logon form and it is passed to the database for verification.

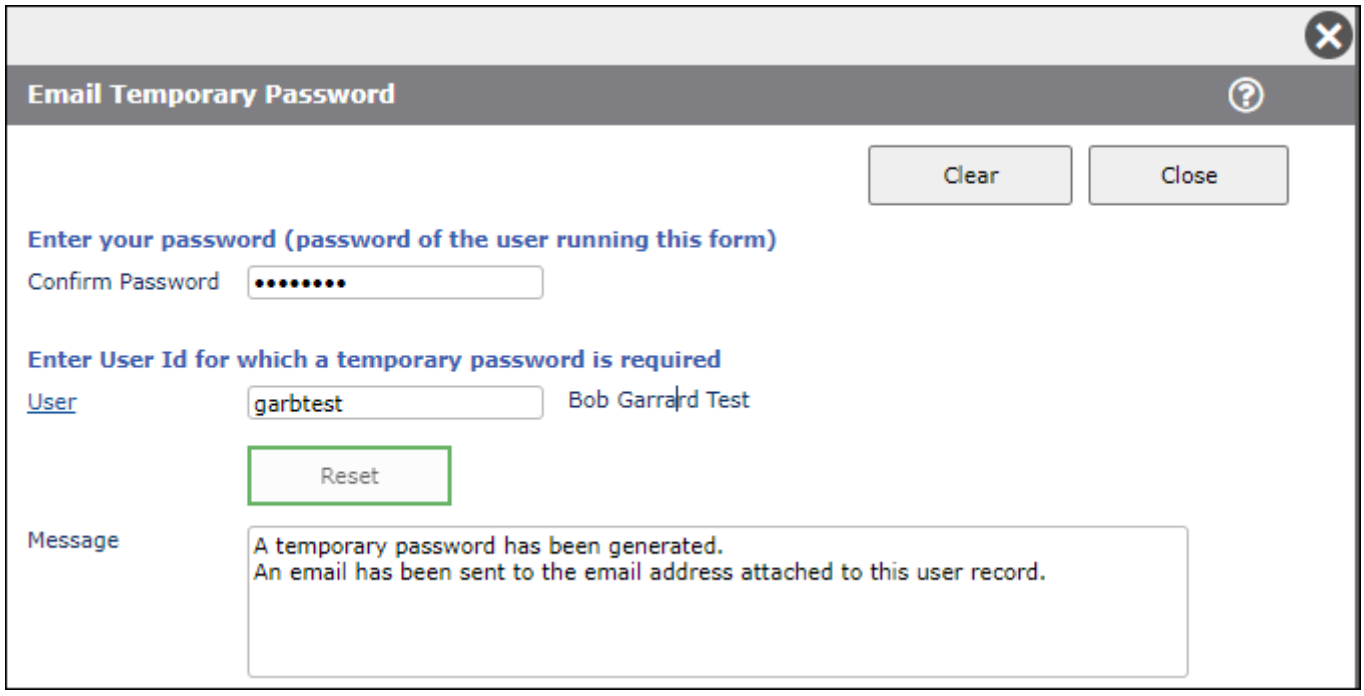
DesignBais User Security Procedure

The following notes describe the way to set up a new DesignBais user or to reset the password of an existing user.

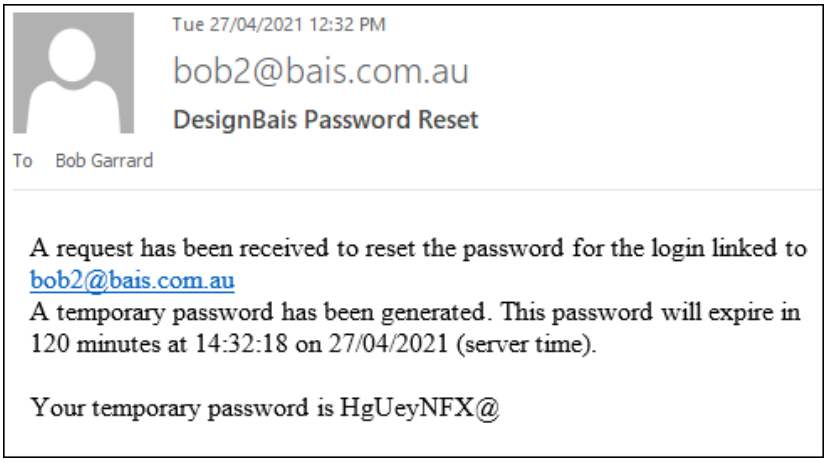
The *Copy User* option creates a copy of an existing user record. The user password on the new record is set to null. The email address on the new record must be updated.

The System Administrator can then use the *Email Temporary Password* option to send a temporary password to the email address on the new user record, or, if a user requests a new password this same procedure can be used to email a temporary password.





The email format appears like this.



The user can then proceed to login to the system. The DesignBais Login form displays.

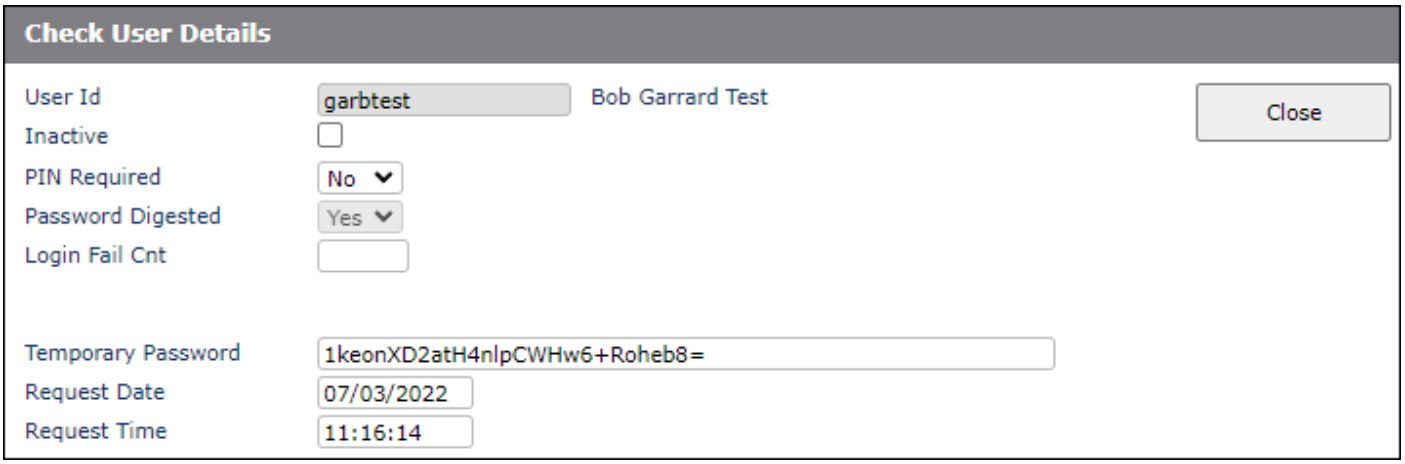
Enter Username and Password. The password to use is the temporary password from the email. This password will expire after a period. If that occurs a new temporary password will be required.

The user is then prompted to change the password.

Providing the *New Password* and *Confirm Password* match then the user's record is updated with the new password and the user is logged into the system.

If a user fails to login correctly multiple times then the system will introduce a delay of increasing length between attempts to login. This is to combat unfriendly attempts to guess a user password.

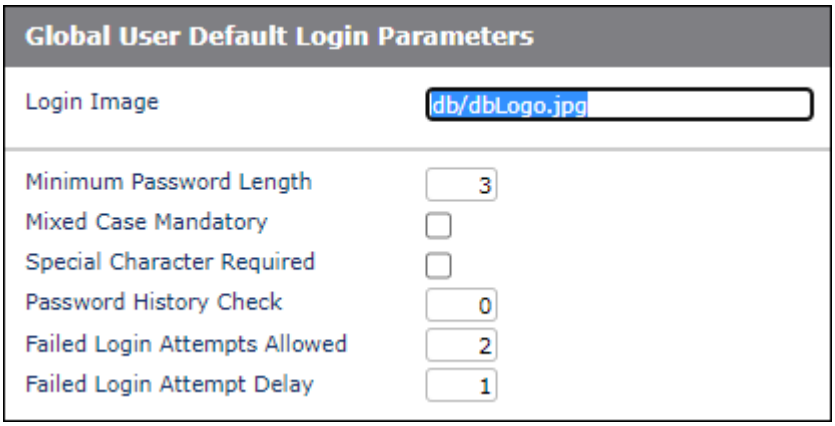
The System Administrator can view the details of the user record using the *User Details* button on the Actions dropdown in User Maintenance. In the example below the user *garbtest* has been issued a temporary password. The plain text temporary password is emailed to the user and stored in digested form on the system. Entering the digested password in the login form will not gain entry.

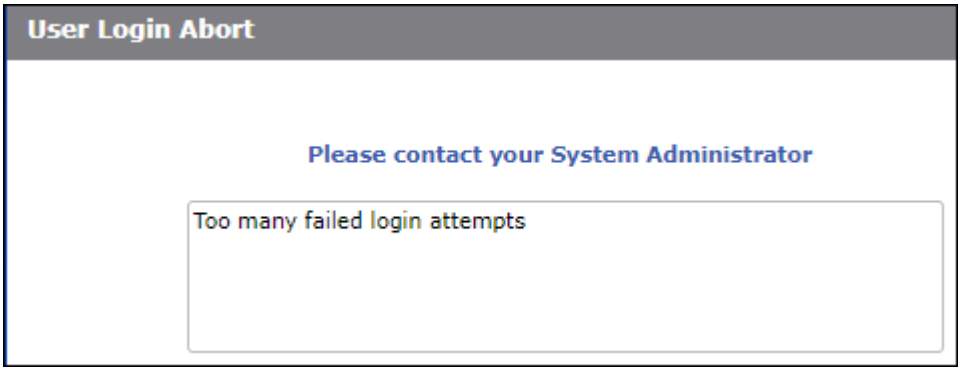


The Global Login Parameters are documented elsewhere, however the relevant parameters are shown here:

The System Adminstrator can set the number of *Failed Login Attempts*. If a user does not enter a valid combination of *Username* and *Password* after this number of attempts then the login process will be aborted and the User Login Abort form will ne displayed.

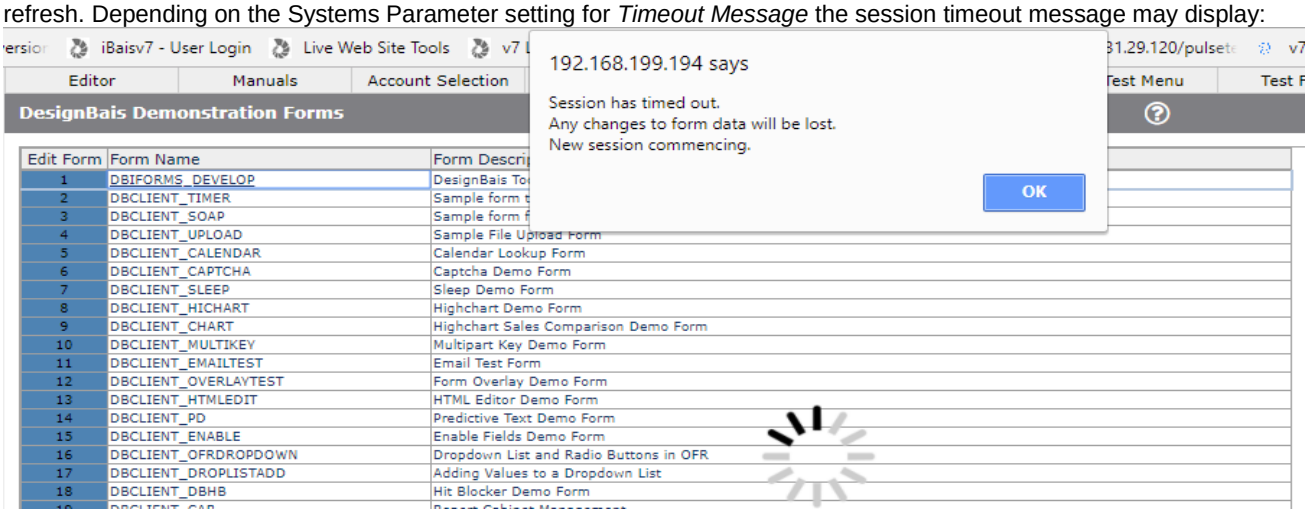
The *Failed Login Attempt Delay* is the number of seconds delay after the *Login* button is clicked. This delay is designed to prevent repeated rapid attempts to break into the system.





DesignBais Page Refresh

A DesignBais session will remain active, following the most recent server hit, for the number of minutes defined in the *Session Timeout* field in System Parameters. If a user attempts to re-activate the session after that period of time then there will be a page refresh. Depending on the Systems Parameter setting for *Timeout Message* the session timeout message may display:



When starting a new session in this scenario, prior to the timeout period, DesignBais will extract the session user from the old session and determine the start account and start form for that user. The new session will re-open in the user's start form in their start account.

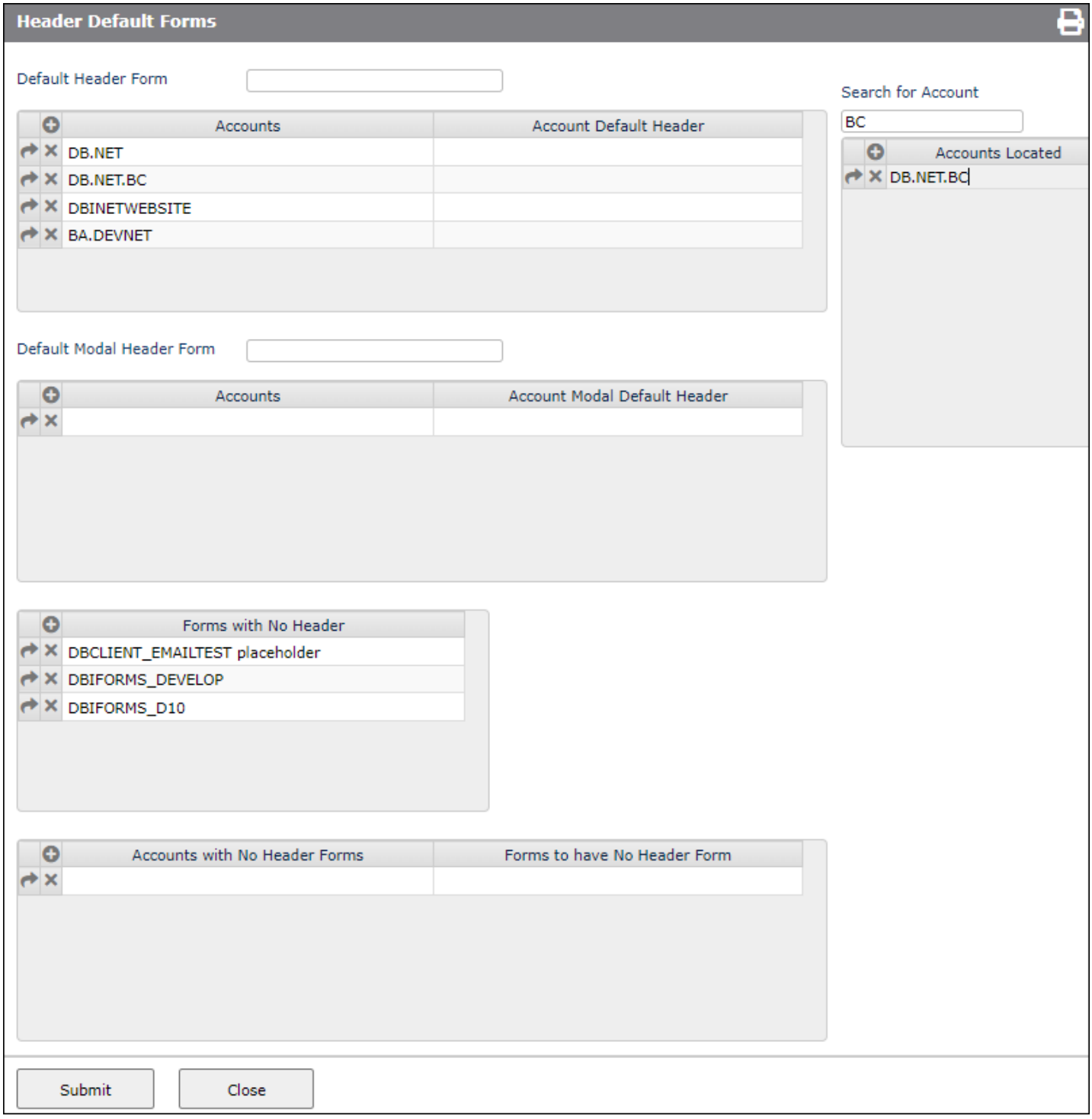
Prior to the timeout period having elapsed a user initiated Page Refresh will pick up the last user logged in. Therefore automatic refresh of an existing validated user will happen if you press F5 (click browser refresh), ctrl+F5 or press a browser back button to arrive back in DesignBais after being elsewhere.

This mechanism prevents re-authentication for applications that implement their own login form.

Page Refresh with automatic assignment of a validated user will **not** occur after the Timeout Period has elapsed. User re-authentication will be required.

Note that after a session has timed out any changes to the form that was active before the timeout will be lost, regardless of whether the timeout message is displayed.

Header Default Forms (GLOBALHEADER)



If this DBIGLOBAL record GLOBALHEADER exists, it is used to provide the form name that will be inserted as the default header. Note the use of “place holder” in order to temporarily disable the use of a header or a form. The fields on this form are not validated so any literal can be added in order to achieve this and retain an audit trail that makes re-instatement at a later date easier.

Default Header Form

The Global Header Default Form is applied to any non-modal form, that has no Header Form specified, when it is run in an Account without its own Default Header.

Note that if the an account name is present in the list of Accounts (below), and no form is entered in the associated Account Default Header column then this will serve to suppress the application of the Global Header Default Form in that account. This serves as a means of defining a Global Header Form and turning it off and on, per account.

Form Name is entered as FILENAME_FORMNAME.

Accounts

Enter the name(s) of any Account(s) in which a specified Default Header Form is to be applied to non-modal Forms that have no Header Form when run in the nominated Account.

Account Default Header

The Default Header form to be applied in the nominated Account to non-Modal forms. May be left blank if no Default Header Form is required. Form Name is entered as FILENAME_FORMNAME.

Search for Account

Enter a search string to check whether a particular account is in the Accounts list. This useful on large systems where there is a long list of accounts.

Default Modal Header Form

The Global Header Default Form is applied to any modal form, that has no Header Form specified, when it is run in an Account without its own Default Header.

Form Name is entered as FILENAME_FORMNAME.

Accounts

Enter the name(s) of any Account(s) in which a specified Default Header Form is to be applied to modal Forms that have no Header Form when run in the nominated Account.

Account Modal Default Header

The Default Header form to be applied in the nominated Account to Modal forms. May be left blank if no Default Modal Header Form is required. Form Name is entered as FILENAME_FORMNAME.

Forms with No Header

Forms that may be run in any Account that will not have the Default Headers applied i.e. they will have No Header Form. The list may include modal or non-modal forms. Form Name is entered as FILENAME_FORMNAME.

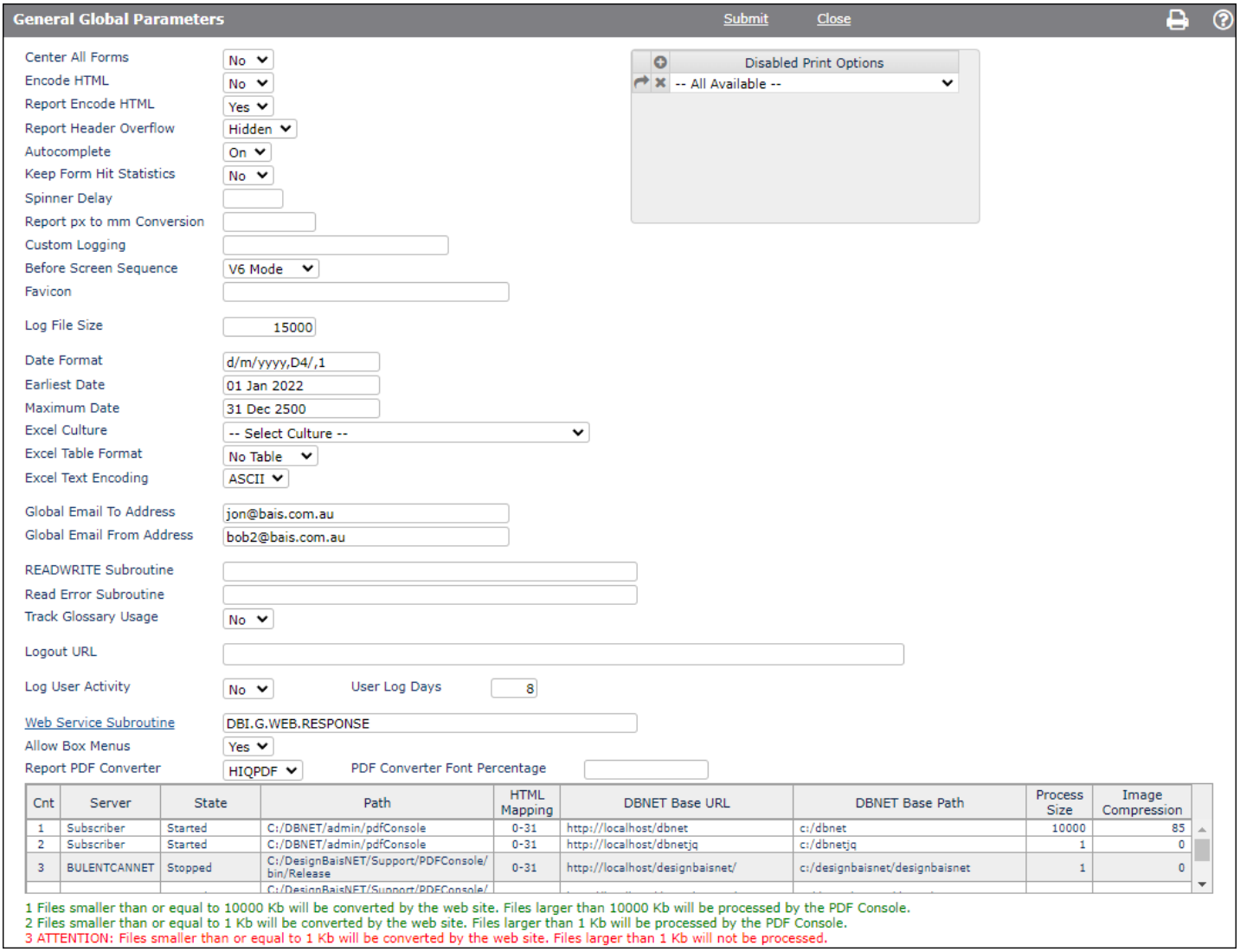
Forms to have No Header Form

Enter the forms that will not have the default header applied when the form is run in the nominated Account. May be modal or non-Modal. Form Name is entered as FILENAME_FORMNAME..

Footer Default Forms (GLOBALFOOTER)

If this DBIGLOBAL record GLOBALFOOTER exists, it is used to provide the form name that will be inserted as the default Footer. It has the same format as GLOBALHEADER.

General Parameters



Center All Forms

Forms with Form Centered set to "Inherit" will use the System Parameter Center All Forms setting unless it too is set to "Inherit" in which case this Global Setting will be applied. Select "Yes" to indicate that all "Inherit" forms are to be centered. "No" is the default.

Stored in the DBIGLOBAL record GENERAL.

Encode HTML

Output fields containing HTML elements should be encoded to prevent XSS injection attacks. Encoding HTML means converting characters that have meaning in HTML to their display only string e.g. < is encoded as & l t ; (without the spaces).

This means that any HTML will display as entered but will not be active in the browser.

Encoding may be set to Inherit, Yes or No.

Encoding may be set on Output Fields. If an Output Field is set to Inherit (the default action) then the System Parameter setting will be applied, if the System Parameter value is Inherit then this Global Setting will be applied.

The Global Setting will default to No encoding to provide backwards compatibility for existing applications.

It is recommended that HTML Encoding be turned on with only fields requiring active HTML elements having No encoding.

Stored in the GENERAL Global Parameters record.

Report Encode HTML

This setting applies to reports built in the Report Designer only. Use the Encode HTML for Form elements.

Output fields containing HTML elements should be encoded to prevent XSS injection attacks. Encoding HTML means converting characters that have meaning in HTML to their display only string e.g. < is encoded as & l t ; (without the spaces).

This means that any HTML will display as entered but will not be active in the browser.

Encoding may be set to Inherit, Yes or No.

Encoding may be set on Output Fields. If an Output Field is set to Inherit (the default action) then the System Parameter setting will be applied, if the System Parameter value is Inherit then this Global Setting will be applied.

The Global Setting will default to Yes (HTML encoding) to provide backwards compatibility for existing applications.

It is recommended that Report HTML Encoding be turned on with only fields requiring active HTML elements having No encoding.

Stored in the GENERAL Global Parameters record.

Report Header Overflow

Overflow is hidden in report fields by default to prevent text from overlapping. You can set this field to to allow overflow in the header, column header and footer sections of reports. The setting may be overwritten in System Parameters.

Autocomplete

This setting allows you to set autocomplete off for all forms. This will apply for browsers that support this <form> attribute.
Stored in the DBIGLOBAL record GENERAL

Keep Form Hit Statistics

Count of the number of times a form is run. Statistics are stored on the DBISTATS file with record id FORMHIT*filename*formname. The total number of times the form is used is based on how many times the routine DBI.G.GSNET is called, divided by 2 (DBI.G.GSNET is called twice per form use).
This parameter is stored on the DBIGLOBAL record with id GENERAL.

Spinner Delay

This setting allows you to set a delay before the DesignBais event spinner is invoked. This will stop the display of the loader for short database events. Enter a number of milliseconds e.g. 2000 if you want to give the database 2 seconds to respond before the spinner is displayed.

Report px to mm Conversion

DesignBais Version 6 used a factor of 4.05 to convert px to mm. This was changed to 3.7795 for Release 7 and 8. You can set a different value by entry in this field. The default remains as 3.7795. This global value can be overridden for a particular account in the System Parameters.

Custom Logging

Enter the name of a subroutine that will be used to provide your logging. Place the entry point in attribute 1 of the subroutine argument. Custom parameters may be added following a pipe (|) character. Refer to the Custom Logging entry in System Parameters.

Before Screen Sequence

This setting will control when the BEFORE SCREEN subroutine attached to the menu structure is run. A setting here will override the Global setting.

Leave as *Inherit* if this account is to use the Global Parameters Setting.

Menu Only the BEFORE SCREEN will only be called when a menu option is clicked. It will be before the form BEFORE DISPLAY is invoked.

	First	the BEFORE SCREEN will be invoked whenever a new form opens (menu click or PROCESS.STACK). It will be called before the form BEFORE.DISPLAY.
	V6 Mode	the BEFORE SCREEN will be invoked whenever a new form opens (menu click or PROCESS.STACK). It will be called after the form BEFORE.DISPLAY. This was the V6 behaviour.

Phantom Command

On D3 only the phantom command can be specified. The default value is “z”. If the alternative “zs” or “zsd” is required then enter the value in this field.

Favicon

The system favicon appears in the browser tab. Enter the path to the favicon. If left blank the default is the DesignBais favicon located at favicon/favicon.ico.

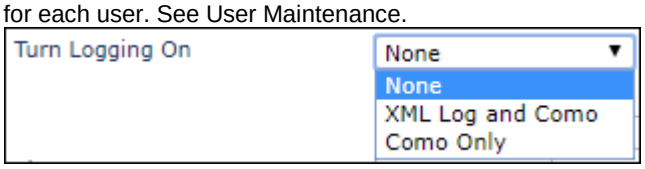
Disabled Print Options

List of Print Options NOT to be offered.

Log File Size

The size of the log file in bytes when user logging is active. This feature is turned on when a record named LOGSIZE exists in file DBIGLOBAL. The default size is 10K bytes. If record LOGSIZE does not exist the user como record will be effectively empty.

If the size of the log record is 25% greater than the trim size requested, it is then trimmed.
The 25% allowance stops the log being trimmed on every server hit. Note that logging must be turned on if required for each user. See User Maintenance.



Stored in the DBIGLOBAL record LOGSIZE.

Date Format

There is an option to define the date format for all accounts via this field in DBIGLOBAL. Use this field to do this. Note that the date format defined here will be overridden by an entry in the Date Format in the System Parameters in any account or by the Date Format in a user record.

This field, if populated, is used to ensure that date fields are interpreted correctly by the database server (in any account where System Parameters does not override).

Refer to the System Parameters Date Format for detailed explanation.

Earliest Date

The earliest date to be entered into the application.
May be a valid date or a calculation like T, T+nnn or T-nnn where T is the date at run time plus or minus a number of days.
This value may be overridden by the value in System Parameters.
This is useful to prevent keying errors in the year that produce a 5 digit year number that is not handled correctly by a D4 date conversion or a date too far back in time.

Maximum Date

The maximum date to be entered into the application.
May be a valid date or a calculation like T, T+nnn or T-nnn where T is the date at run time plus or minus a number of days.
This value may be overridden by the value in System Parameters.
This is useful to prevent keying errors in the year that produce a 5 digit year number that is not handled correctly by a D4 date conversion or a date too far back in time.

Excel Culture

The Excel Culture is used by the conversion to Excel component to determine if an exported value is a date. The output format is determined by Date Format parameter.The user setting if present, overrides the System Parameter which in turn, overrides the Global setting.

Excel Table Format

This setting determines if the data exported to Excel is to be displayed in table format and if so, the style of table. The Global setting is overridden by the the System Parameters setting which in turn is overridden by the user setting. If No Table Format is selected then this will mean that the raw data is exported to Excel with no table formatting.

Excel Text Encoding

This setting determines if the data exported to Excel is to be encoded as ASCII (DEFAULT) or as utf-8. Using UTF8 together with the Culture setting allows Currency symbols to be applied when the Excel file is produced. May be overridden by the User setting. May be overridden by the User or System Parameter setting.

Global Email To Address

The email address to which DesignBais licencing errors will be sent.

Global Email From Address

The email address that will be used as the sender address for DesignBais licencing errors.

These 2 email addresses are held in attribute 1 and 2 of the record GLOBAL.EMAIL in the DBIGLOBAL file.

READWRITE Subroutine

All DesignBais reads or writes will call the nominated subroutine after completion. The subroutine will be called with the following parameters:

PROCESS.EVENT = “BEFORE READWRITE” or “AFTER READWRITE”
PROCESS.EVENTSOURCE = Name of the file being accessed
PROCESS.PARAMETER = Record id : VM : Process Type

The subroutine name defined in this record will be invoked for the entire application.
It is held in attribute 1 of the record READWRITE in the DBIGLOBAL file.

Read Error Subroutine

When a DesignBais read of a sessions file fails execution is stopped with a message displayed to the user. This READ ... ON ERROR process has been introduced as some databases may not crash if a READ fails which may result in application errors.

If a subroutine name exists in this field then the subroutine will be called before execution stops:

PROCESS.EVENT = "READERROR"
PROCESS.PARAMETER = Key:VM:Filename

This routine allows the application to log the read errors so that issues with the sessions files may be monitored.
The subroutine name defined in this record will be invoked for the entire application.
It is held in attribute 1 of the record READERROR in the DBIGLOBAL file.

Track Glossary Usage

The “Where Used” tracking feature of the glossary may be turned of using this field. Where a text string is used is tracked by default in "Glossary Maintenance" via the "Where Used" multivalue. This may now be switched off either Globally or in the System Parameters.

Logout URL

When the user clicks the logout button there may be a requirement to divert them to another web page (URL). The URL defined in this record will provide the default logout url for the entire application.

The standard DesignBais logout url is: http:// {ip address} /db/loggedout.htm

It is held in attribute 1 of the record LOGOUT.URL in the DBIGLOBAL file.

Log User Activity

This User Activity may be logged on the file DBIUSERLOG and reported. This may be set at System or User level as well. Users with Log User Activity set to "Inherit" with use the System Parameters value. Systems with Log User Activity set to "Inherit" will use the Global Parameter. Select "No" to stop logging user activity in the DBIUSERLOG file. "Yes" is the default. Stored in the DBIGLOBAL record USERLOG.

User Log Days

The number of days to keep DBIUSERLOG records.

Web Service Subroutine

The subroutine name defined in this record will be invoked for the entire application. It is held in attribute 1 of the record WEBSERVICE in the DBIGLOBAL file. It is a two argument subroutine used to provide web services from within DesignBais applications. See DBINET DBI.G.WEB.RESPONSE for an example.

Report PDF Converter

DesignBais Reports may be converted to PDF files. There are two tools implemented in DesignBais for this purpose - ABCPDF and HIQPDF. The new HIQPDF allows the conversion of larger report HTML files and is the default process. The original ABCPDF may be selected if preferred. The selected option is stored in the DBIGLOBAL record REP2PDF.

PDF Converter Font Percentage

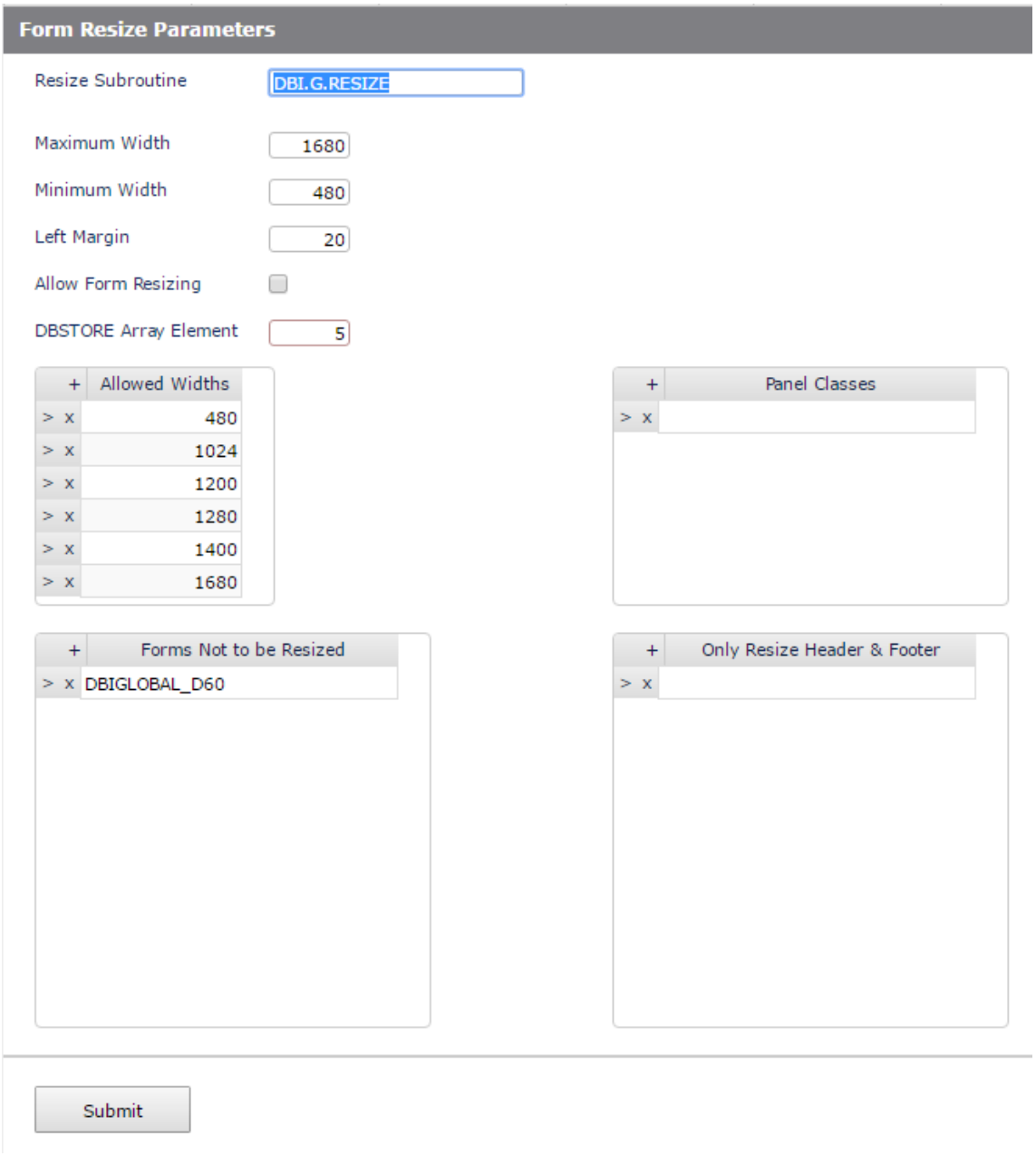
DesignBais Reports may be converted to PDF files. The conversion process may truncate text in some fonts that is viewable in HTML when the form is constructed. To overcome this use this scaling factor to produce a separate stylesheet "DesignBaisPDF.css" that is included with the HTML to be converted rather than "DesignBaisStyle.css" or "DesignBaisPrint.css". Stored in the DBIGLOBAL record REP2PDF. No value or a value of 100% will result in using the standard DesignBaisPrint.css stylesheet.

The PDF converter, like browsers, renders fonts at different sizes. In order to save the developer having to size reports to suit the PDF conversion DesignBais introduced this conversion factor.

After setting the percentage (97% is commonly a good starting point) you will need to dummy amend a Style to generate the DesignBaisPDF.css in the website and then refresh your browser. Any subsequent PDF conversions should see a reduction in the font size.

There may still have cases where the report may need to be adjusted. In addition the report may render differently in other browsers such as FireFox. In general it is good practice to set field col spans to allow room for the font to be larger than the browser you are in.

Form Resizing



This function will dynamically resize form elements as the browser window size is changed.

Resize Subroutine	A subroutine with two arguments to do the resizing of form elements. Refer to DBI.G.RESIZE. 1) CALLTYPE: can be either RESIZE, LOAD or DBADDFRAME 2) RESIZE.PARAMS holds the DBIGLOBAL "RESIZE" record which is read before calling the subroutine.
Maximum Width	Maximum width in pixels
Minimum Width	Minimum Width in Pixels
Left Margin	Left Margin when resizing in Pixels
Allow Form Resizing	Form Resizing may be switched off.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

DBSTORE Array Element DBSTORE array element to hold previous resize maximum columns.

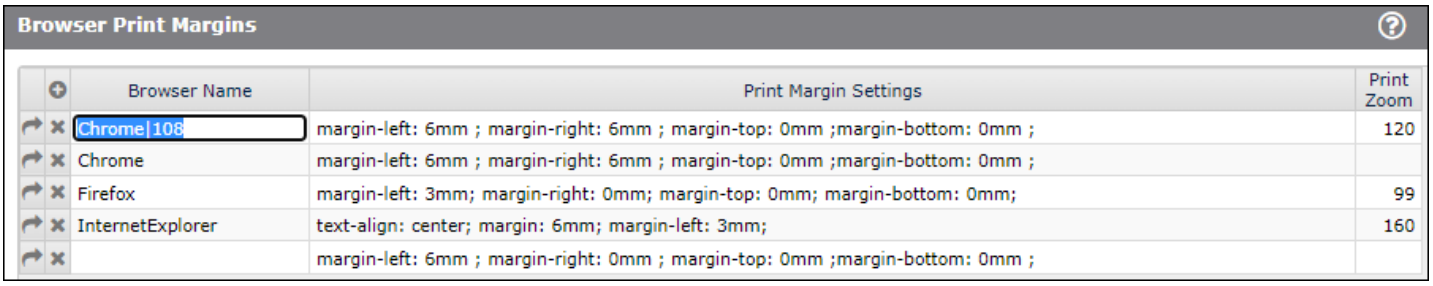
Allowed Widths Allowed widths in Pixels e.g. 1024, 1200, 1280 & 1400

Panel Classes Classes that indicate Panels - not implemented

Forms Not to be Resized Forms NOT to be resized.

Only Resize Header & Footer
Forms where only the Header and Footer Fields will be affected by resizing.

Browser Print Margins



Browser Name

The name of the browser as used in the DesignBais common variable DBW3CBROWSERVERSION. Leave this field blank if the associated Print Margin Settings in the next column are to be used as the default print margin style attributes.

Example: Chrome or Firefox

Print Margin Settings

Enter style attributes to be included in the style @page tag.

Example: margin-left: 6mm ; margin-right: 0mm ; margin-top: 0mm ;margin-bottom: 0mm ;

Print Zoom

This parameter only applies to Internet Explorer. Internet Explorer applies the default "Enable Shrink to Fit" before landscape pages are rotated for printing. If "Enable Shrink to Fit" is left checked then, before rotation, the page is shrunk to fit the page's long margin to the portrait width. This leaves a small printed report. So the *Print Zoom* factor is applied to make the page larger before it is shrunk.

The page needs to be expanded by a percentage enlargement factor of around 160.

If "Enable Shrink to Fit" is switched off by the user than this factor is not required.

The setting here is the Global Default which may be overwritten on the Users record by entering a value.

Entering zero in the Users record, as opposed to leaving a null value, will cause DesignBais to ignore the *Print Zoom* setting altogether.

Browser Print Rotation

From release 7.2.2.18 onwards DesignBais has implemented print rotation in order to overcome the issue that not all browsers interpret the *@page* CSS at-rule.

The @page CSS at-rule is used to modify some CSS properties when printing a document. You can't change all CSS properties with @page.

DesignBais v7.2.2.18+ now sets the page size for portrait. Width is set as the lower number in the report dimensions. If required DesignBais rotates each page of the report. A "dbaisrot" style and appropriate items are added to the report HTML as needed.

Users should now set their default printer orientation to portrait to take advantage of this feature.

For the more technical user the following expands on the approach DesignBais has taken to printing. For a standard page DesignBais adds PAGESIZE parameters to the stored report file and an @PAGE command to the report HTML. This gives:

```
@page{size: 210mm 297mm;text-align: center; margin: 0mm; margin-left:3mm;}
```

Not all browsers support the method of swapping the size parameters (width v height) in order to allow a landscape page to be printed with the browser print orientation set to portrait.

To overcome this problem DesignBais now uses the css transform (with no prefixes). See https://www.w3schools.com/cssref/css3_pr_transform.asp :

Each report page is wrapped in a * * which has its content rotated, translated and scaled.

For now the scaling factor is 0.95. It was found that 0.99 works but does not allow for variation in the browser settings for *Page Setup Headers and Footers*.

DesignBais style settings to set page size, which may or may not be used by a browser, allow for margins to be parameterised:

```
@page{size: 210mm 297mm;text-align: center; margin: 0mm; margin-left:3mm;}
@media print {html, body {width: 210mm; height: 297mm;  text-align: center; margin: 0mm; margin-
left:2mm;
```

DesignBais applies the following class to a * * that wraps each page:

```
.dbaisrot {transform: rotate(270deg) translate(-297mm, 0mm) scale(0.99,0.99);transform-origin: 0 0;}
@media screen {.dbaisrot {padding-bottom: 87mm;}}
```

For landscape printing DesignBais wraps each report page in a * * which is then rotated to allow for output by a standard portrait print configuration.

Internet Explorer applies the *Enable Shrink To Fit* before the page has been rotated. This results in the landscape width (297mm) being reduced to portrait size (210mm).

To overcome this DesignBais has added a style *Print Zoom* with a percentage factor to enlarge the output by 160%. If the browser heading and footers are turned off the factor can be increased slightly.

To allow for the default header & footer DesignBais sets the InternetExplorer margins to 6mm.

PDF Conversion and PDFConsole

Refer to the Web Component Manual section titled “PDF Document Generation”.

If the PDFConsole is running then the <pdfProcessSize> node controls where PDF conversion jobs are processed. As detailed in the Web Component Manual you can set up the *DBNET/admin/db.config* file *<setup> node* (where DBNET is the name of your website) with this entry:

```
<pdfProcessSize>150000</pdfProcessSize>
```

This way, files larger than 150KB will be proccessed by the PDFConsole and small files will still be processed on the web server. The console can be started from your website admin folder *DBNET/admin/pdfcolsole/pdfconsole.exe*

The PDF Conversion may get stuck with a progress message similar to “Processing … 10%”. This indicates that the PDF Console is not running.

Important Note

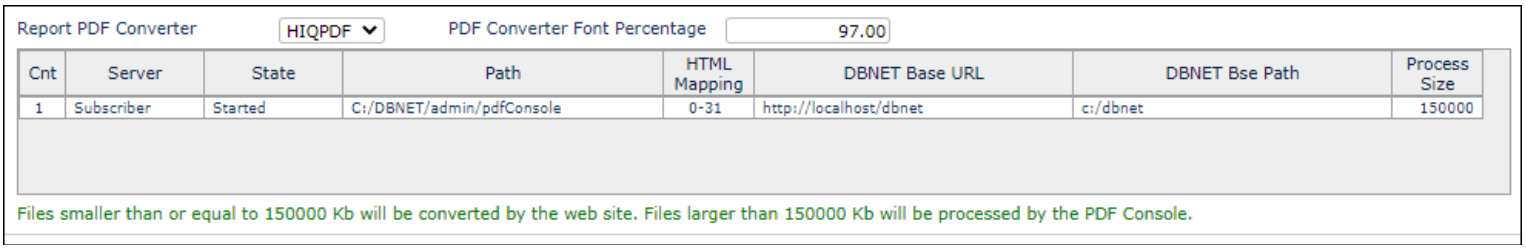
The developer or System Administrator must make sure that **either** the PDFConsole is running or pdfProcessSize is set to -1.

```
<pdfProcessSize>N</pdfProcessSize>
```

- N = -1 then IIS handles all PDF conversions
- N > -1 then any job larger than N is processed by the PDFConsole

For example if N = 10000, any job that is larger than 10KB is pushed to the PDFConsole on the assumption that it is running. **If the PDFConsole is not running then the jobs larger than 10KB are not processed at all**.

The General Parameters option under Global Parameters on the DesignBais Tools side menu includes the setting to determine which PDF Converter to use. If using HIQPDF then the following status display is available. The *State* column indicates state of the converter: Started, Paused or Stopped.



There is node in the *admin/db.config* file for the website to define the required compression level. Use this setting to reduce the size of the PDF files produced by DesignBais.

```
<HIQImageCompressionLevel>0</HIQImageCompressionLevel>
```

- 0 - no image compression during HTML to PDF conversion
- 100 - maximum compression
- or any integer in between

A setting of 85 has been seen to reduce files from 16MB down to 4MB with no significant degredation in resolution.

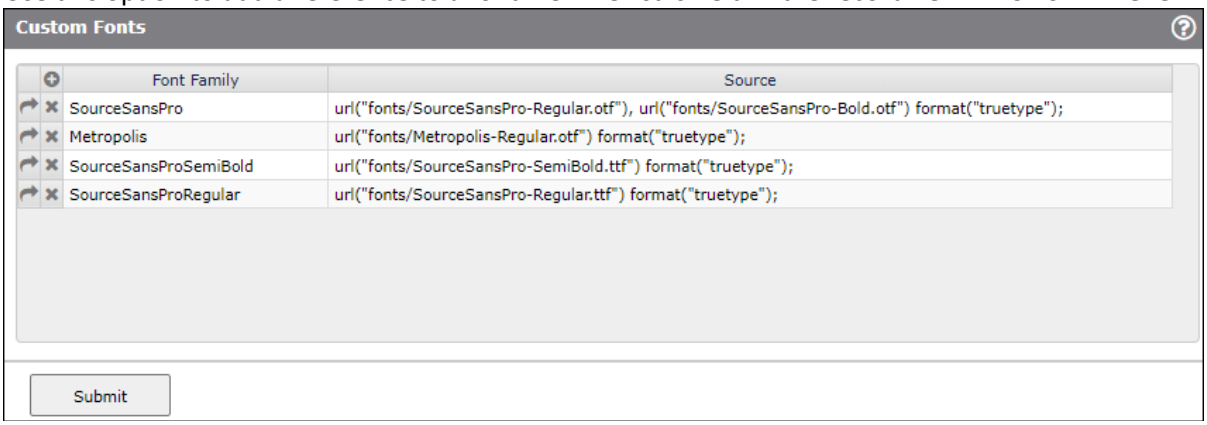
Files to be processed by the PDFConsole are written to the *uploadsin* folder in the website with a name constructed as follows:

filenameMy.html

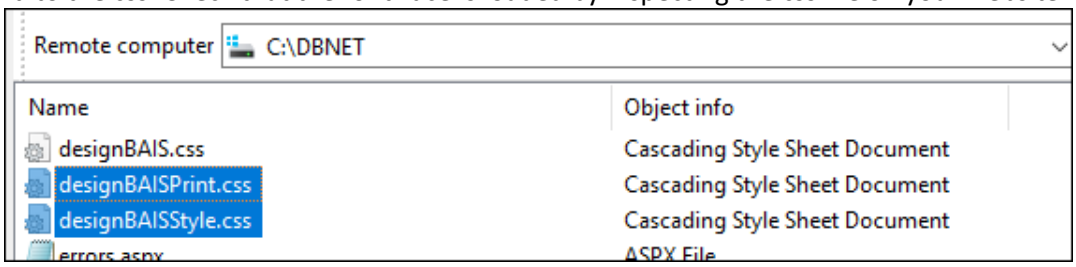
- where M is "p" or "l" denoting portrait or landscape respectively
- and y is a random letter in the range [a-z]

Custom Fonts

Use this option to add a reference to a font file. The list is held in the record *FONTFACE* on DBIGLOBAL.



Dummy amend a style to trigger the re-build of the DesignBais style sheet. This will add the new custom font-face command to the css. Check that the font-face is loaded by inspecting the css file on your website:

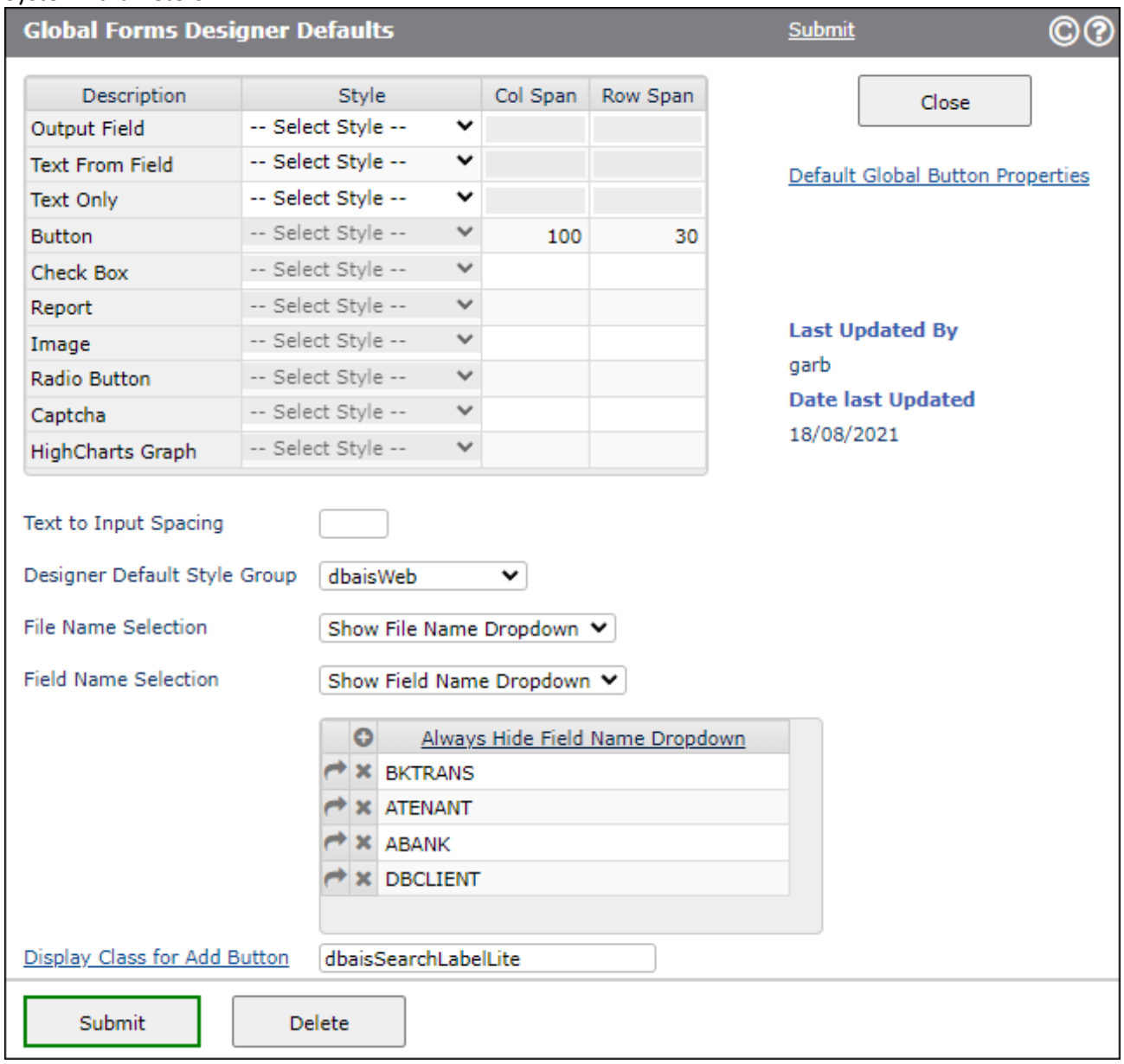


DesignBaisStyle.css

```
@font-face {
font-family: "Code 128";
src: url("fonts/code128.woff2") format("woff2");
font-weight:normal;
font-style:normal;
}
```

Designer Defaults

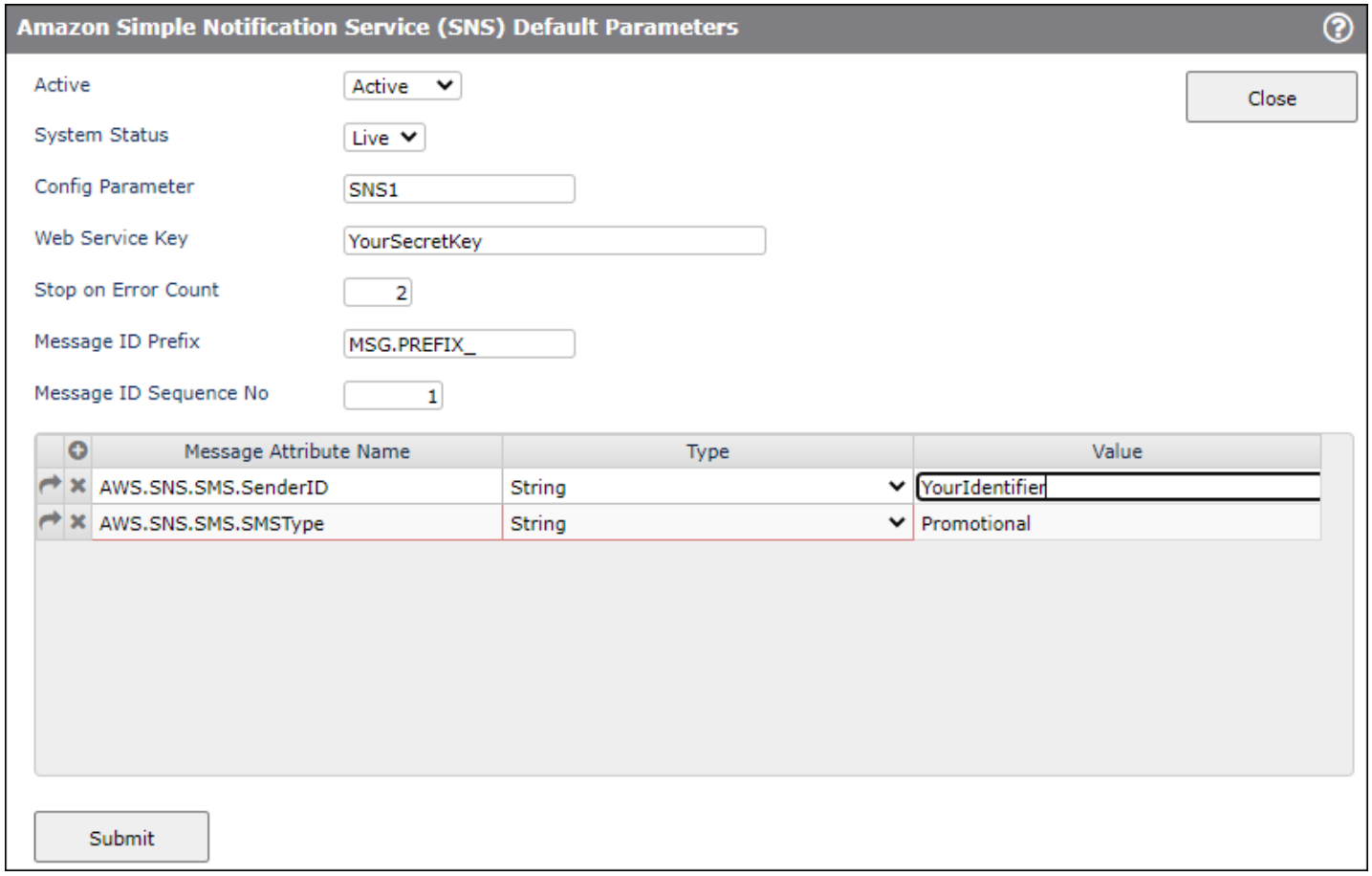
These forms designer defaults apply globally. They are overridden by System Parameter settings. For details refer to System Parameters.



Amazon SNS

DesignBais provides the ability for developers to utilise the Amazon Simple Notification Service (SNS). Amazon Simple Notification Service (SNS) is a notification service provided as part of Amazon web services. It provides a low-cost infrastructure for the mass delivery of messages, predominantly to mobile users.

Amazon Simple Notification Service (SNS) Default Parameters are used by DesignBais to populate any null arguments passed into the DBI.G.SENDSNS subroutine. The parameters are held on DBIGLOBAL with Record Id 'SNS'. Note that these parameters can also be entered for each account where thay are held on the DBIPARMS file with Record Id 'SNS'. Refer to the **System Parameters** section of this Reference Manual.



Active	Active or Inactive. The flag value is Y or N where Y indicates Active. This setting is only visible on the System Parameters settings. If System Parameters settings exist but the flag is set to Inactive (N) then DesignBais will use the Global settings.
System Status	Either *Test* or *Live*
Config Parameter	The name of the SMSParam in the web server db.config file.
Web Service Key	There must be at least one secret key (webServiceKey) that you provide to get authorisation to use this web service. Otherwise, the web service (that sends SMS) is open to public and anyone can use it. Specified in the db.config file <setup> section. For example: <webServiceKey>YourSecretKey</webServiceKey>
Stop on Error Count	The number of re-tries after which the call to the web service will be abandoned.
Message ID Prefix	The Message ID Prefix is used to identify the account, application or indeed the module that initiated the SMS. It will have the sequence number appended when the message is submitted. This may be overwritten within the program logic. Mandatory.

Message ID Sequence No	The sequence number provides a unique key for the message id. It is concatenated with the message id prefix for logging purposes.
Message Attribute Name	These are defined by the Amazon SNS system. The **SenderID** attribute and an associated value must be specified.
Type	Specify whether the value is a string or a number.
Value	The value to be assigned to the Message Attribute Name. **SenderID** can contain up to 11 alphanumeric characters, including at least one letter and no spaces. Support for sender IDs varies by country. **SMSType** can be either 'Promotional' or 'Transactional'.

Refer to the **DesignBais Subroutines and Standard Forms** section of this Reference Manual which details the use of the subroutine DBI.G.SENDSNS. This section documents the arguments required by this subroutine.

The SNS specifications must be present in the db.config file:

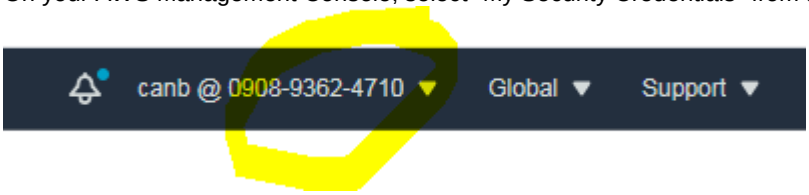
```
<keySpecs>
   <keySpec name="SNS1">
     <SNSKey>AKIAI…….PQ</SNSKey>
     <SNSPass>wR……..zv+KS…..lv+xJ……..G+Y.E15</SNSPass>
     <SNSRegion>ap-southeast-2</SNSRegion>
   </keySpec>
 </keySpecs>
```

Follow these steps to set up Amazon SNS (SMS) functionality in DesignBais:

Step 1: Create an Amazom (AWS) account to sign in to the AWS Management Console
You'll end up with an Account ID, UserID and a password. Enter your AWS management console using those.

Step 2: Create access key and secret key for programmatic calls

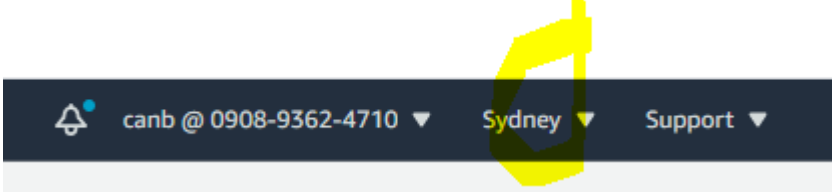
On your AWS Management Console, select "My Security Credentials" from the top left dropdown.



Press "Create Access Key" button which will give you an access key **AK** and a secret key **SK**.

Step 3: Determine your region code

Use the highlighted dropdown to determine your region.



For example: for Sydney use "Asia Pacific (Sydney) ap-southeast-2". The region code is " ap-southeast-2". Determine your region code: **REG**

Step 4: Create other keys

DesignBais requires a web service key for security reasons. Please choose a random string: **YourSecretKey**
DesignBais requires a name for this (SNS) service: Please give this service a name: **SNS1**
DesignBais requires a prefix string for messages: Please choose any string: **MSG.PREFIX_**
DesignBais requires a SenderID for messages: Please choose any string: **Your.Identifier**

Note that we have collected seven variables:

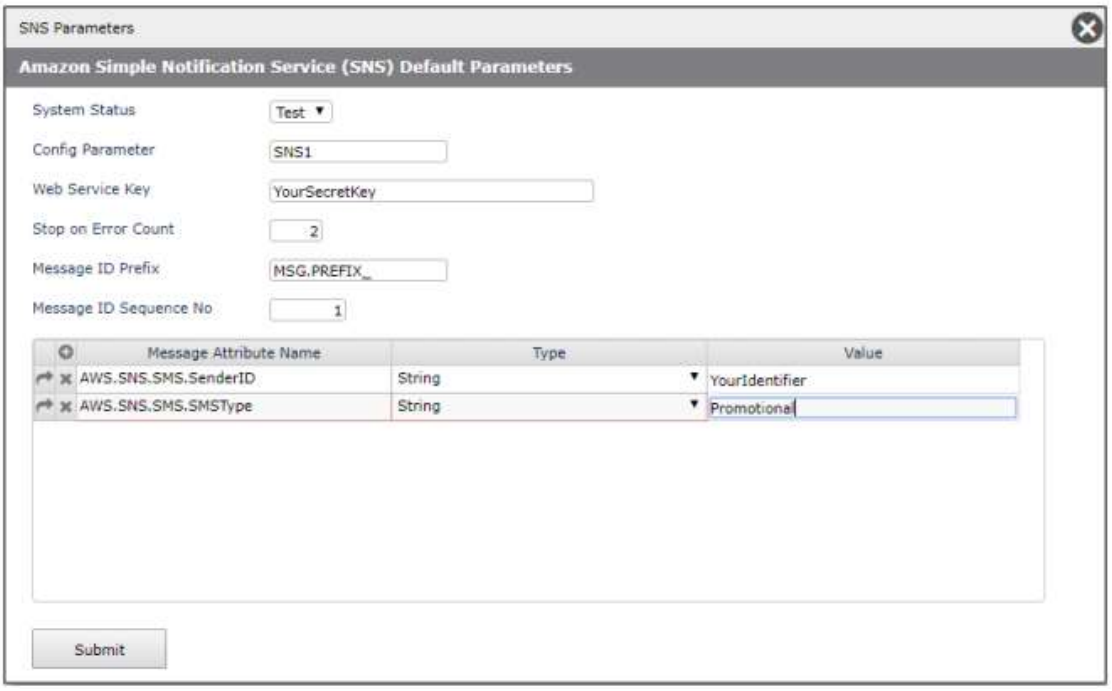
- ***AK, SK AND REG are strings generated/provided by AWS.***
- ***YourSecretKey, SNS1, MSG.PREFIX_, Your.Identifier are strings that YOU generate.***

Step 5: Edit your DBNET/admin/db.config file as follows:

Use the variables determined in previous steps.

```
<dbconfig>

<setup>
    ...
    <webServiceKey> YourSecretKey</webServiceKey>
    ...
</setup>

<keySpecs>
   <keySpec name="SNS1">
     <SNSKey>AK</SNSKey>
     <SNSPass>SK</SNSPass>
     <SNSRegion>REG</SNSRegion>
   </keySpec>
</keySpecs>
...
...
</dbconfig>
```

Step 6: Edit DesignBais System or Global Parameters → Amazon SNS as follows:



- System status: Test does NOT send an SMS but it generates logs. Select Live to actually send SMS.
- Config Parameter: **SNS1**
- Web Service Key : **YourSecretKey**
- Stop on Error Count: Leave this at 2
- Message ID Prefix: **MSG.PREFIX_**
- AWS.SNS.SMS.SenderID: **Your.Identifier**

Notes:

- See DesignBais manual for details on these parameters.
- Set AWS.SNS.SMS.SMSType to Promotional or Transactional. Use "Promotional" during initial development and testing.

Step 7: Program, Test and check logs

Search the manual for the words SNS and SMS:

https://designbais.com/downloads/manual/DesignBais%20Reference%20Manual.pdf

Program and test in accordance with the manual.

Logs are in your DBNET/debug. Three log files *SMS.txt, *SMS_OLD.txt and *SMS_OLD2.txt are provided as truncating logs (each cutting off at 300KB and rotating).

AWS also provides logs and stats if needed.

DBMail

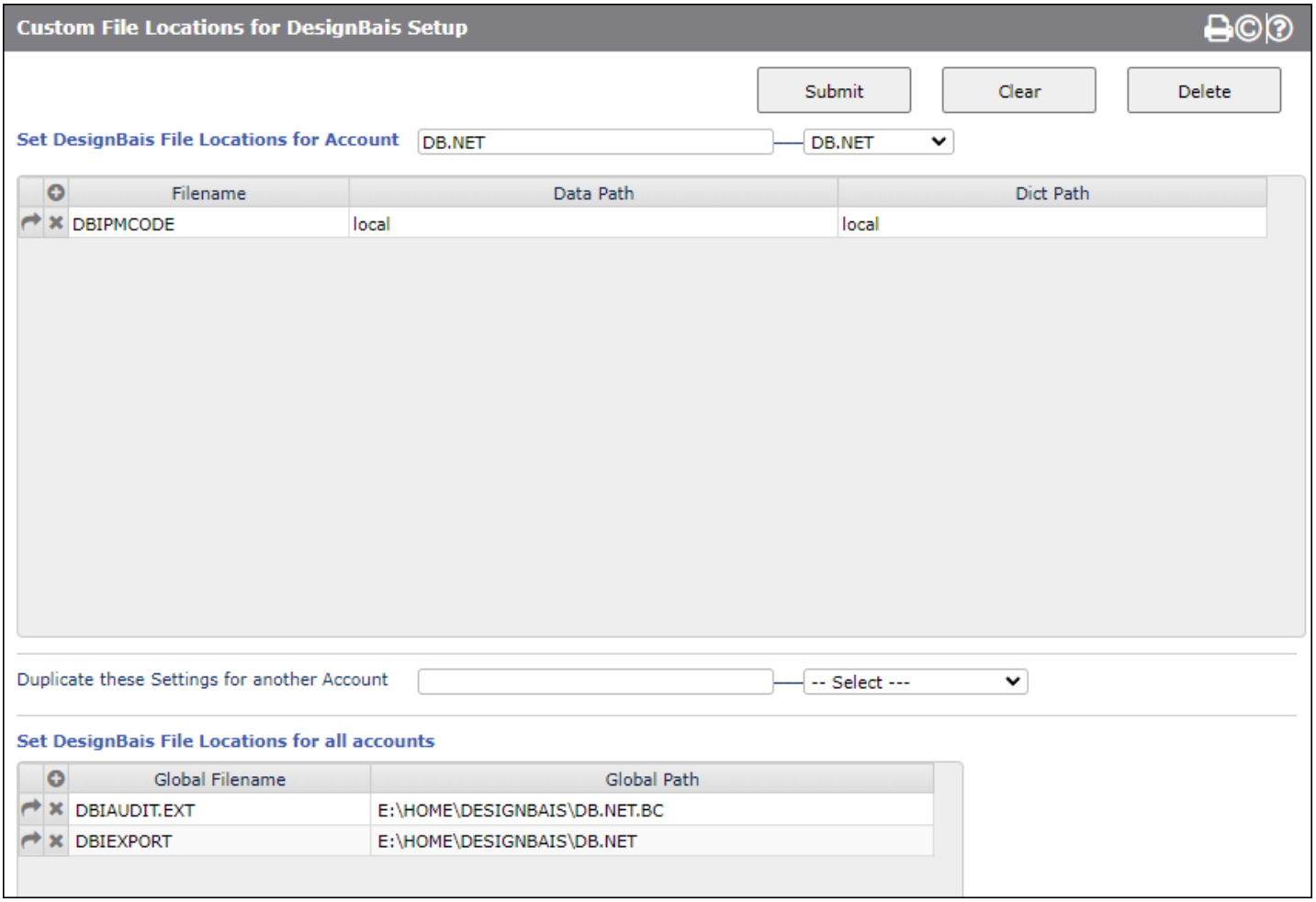
This option opens the DBMail Configuration form.

Refer to the DBMail section of this Reference Manual.

DesignBais File Location

Use this form to set the location of DesignBais files when installing DesignBais in a database account.
By default Designbais will create, or create pointers to, DesignBais files (DBI files) in the DBINET, DBILOGIN and the local account.

If you want a particular DesignBais file, such as DBIAUDIT.EXT, to be used, for example, by all accounts then you can specify the location of the file using this form.



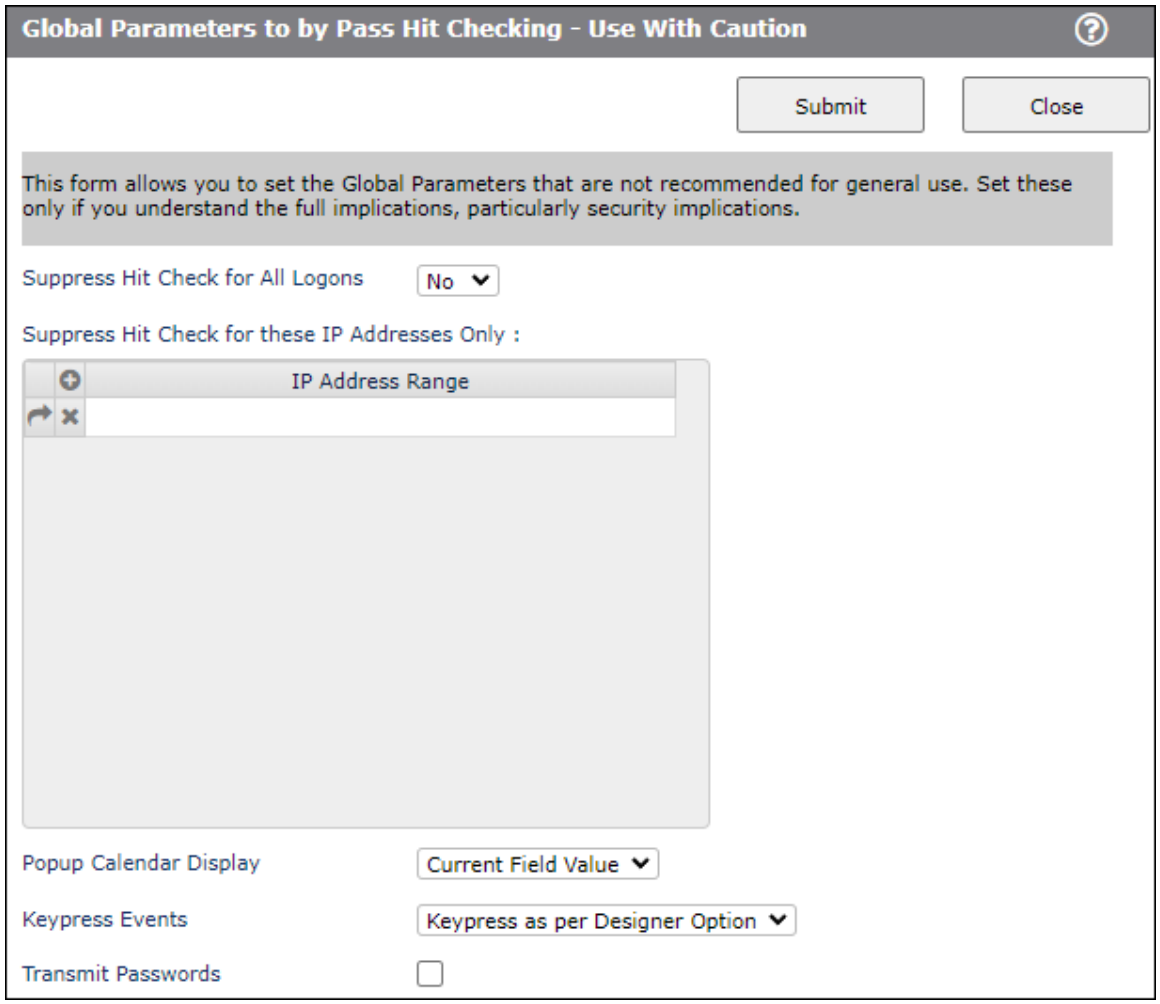
Admin

These parameters are held on record *NOHITCHECK* on DBIGLOBAL.

You must be a member of the DBAdministrator Group to see these options.

Suppress Hit Check for All Logons

This setting must be used with extreme caution. Suppressing the hit checking carried out by DesignBais opens a security hole that can be exploited by malware. This setting should only be set to *Yes* if there are environmental / network issues causing DesignBais hits to arrive out of sequence.



Suppress Hit Check for these IP Addresses Only

The No Hit Check IP Address Range determines what source IP addresses will by pass the standard hit checking.

You can enter a partial IP address, a complete IP address or an IP address range.

If a partial address is entered DesignBais will validate the part entered against the same part of the source IP address.

A sample of an IP Range is 123.456.789.10-50 where the first 3 segments must match the start of the source IP address (123.456.789) and the last segment must be in the range (10 to 50).

IP6 addresses may be used. You must enter either all 8 ":" separated segments or the last 4. If all 8 are present then they will be matched to the source IP otherwise only the last 4 segments of the source IP will be used. Again a range may be indicated by using a "-" in the last segment.

Popup Calendar Display

The Popup Calendar can be set to display either the current date or the date value in the database field.

F – Field Value
D – Current Date

This setting is stored in the DBIPARMS record id 0 (zero).

Keypress Events

The "Force Focus and Keypress" option will include an onfocus event with all keypress fields. The focus event will be a VALIDATE in the database code and the KEYPRESS will not be followed by a VALIDATE. This is also an option in Forms Designer however, this setting will override the normal Keypress handling.

Null – Keypress as per Designer Option
1 – Force Focus and Keypress

Stored on DBIGLOBAL record NOHITCHECK attribute 4.

Transmit Passwords

For security reasons passwords, by default, are not sent to the web component. This means that DesignBais applications cannot display bullets in a password field, although this is the recommended approach for security.

Check this option to allow the transmission of password fields in data component to web component XML packets.

Stored on DBIGLOBAL record NOHITCHECK attribute 5.

Meta Data

The DesignBais Web Component allows custom META tags, CSS and SCRIPTs. Use of scripts are documented in the Web Component manual (WEBCOMP). WEBCOMP and DATACOMP both have advantages depending on the situation. WEBCOMP custom scripts provide access to a number interesting parameters (FORM NAME etc.) and events like (before hit, after hit). The DATACOMP scripts can use data stored in the database.

Script References

If you wish to add any script references to the default DesignBais page you may add them to this record.

The lines entered in the maintenance form are converted to attributes when saved to DBIGLOBAL.

If the SCRIPTS record contains only a single attribute and that attribute begins with an @ sign, a subroutine name is assumed and the routine (defined after the @) will be called with the PROCESS.EVENT set to “LOAD SCRIPTS”.

Eg. @GETSCRIPTS

The scripts definition will then be populated with the content of the DBVALUE common variable as returned by the routine. DBVALUE should be an attribute delimited list of script references.

If the attribute does not commence with the @ sign then the contents of this record will provide the detail. It is stored as a multiattributed record in the DBIGLOBAL file with a record id of SCRIPTS.

The scripts in System Parameters or Global Parameters Meta Data options may now contain:

- dbonload = script to run when the script file is loaded into the form.
- dbname = name to load the script. The default names are DBIACCOUNT:”_scr_”:nn or SCREENROOT:”_scr_”:nn. Eg PRODACC_scr_1 or DBEXEC_A1_scr_1.

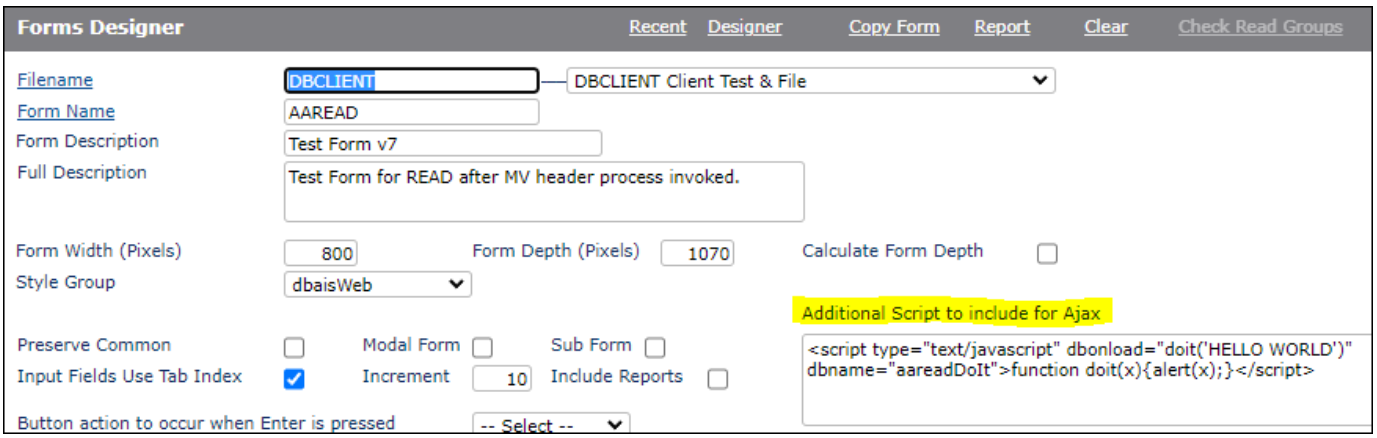
It may be overridden in the System Parameters General maintenance for a particular account.

Example:

```
<script>
(function() {
var i=0;
$(document).ajaxComplete(function() {
i++;
if (i==3){
console.log("Do something at the 3rd stroke");
}
});
})();
</script>
```

This example displays a message on the JS console at the 3rd server hit (in Chrome, press F12, click console TAB and expect this message at the 3rd click). Replace "console.log" with "alert" to see a message box.

From the field in Forms Designer the event is “LOAD FORMSCRIPTS”.



When sending the function:

- THIS.CODE = src attribute if present (for addScript) else it will be the value within the <script></script> tags (for addCode)
- THIS.LOAD = the dbonload value if present else blank (no script will run automatically)
- ID = the dbname value or the default outlined above

addScripts:'("':THIS.CODE:'","':THIS.LOAD:'","':ID:'")'

Meta Tags

This record may contain extra definition for the meta section of the DesignBais form. This allows for code to be added to provide specific instructions for devices or browsers.

The lines entered in the maintenance form are converted to attributes when saved to DBIGLOBAL.

If the META record contains only a single attribute and that attribute begins with an @ sign, a program name is assumed and the progam (defined after the @) will be called. The meta definition will be then filled with the DBVALUE variable.

Eg. @GETMETA

If not the contents of the record will provide the detail.

Stored as a multiattributed record in the DBIGLOBAL file with a record id of META.

Meta tags have many uses such as to provide keywords for search engines. If you load an example in this form, refresh the browser, then open DesignBais you can View Source to see the meta tags.

May be overridden in the System Parameters General maintenance for a particular account.

Custom CSS Links

This record contains any extra link details that you would like to include into the DesignBais load page. This is very useful for including additional references to custom style sheets.

The lines entered in the maintenance form are converted to attributes when saved to DBIGLOBAL.

If the LINKS record contains only a single attribute and that attribute begins with an @ sign, a program name is assumed and the progam (defined after the @) will be called. The links definition will be then filled with the DBVALUE variable.

Eg. @GETLINKS

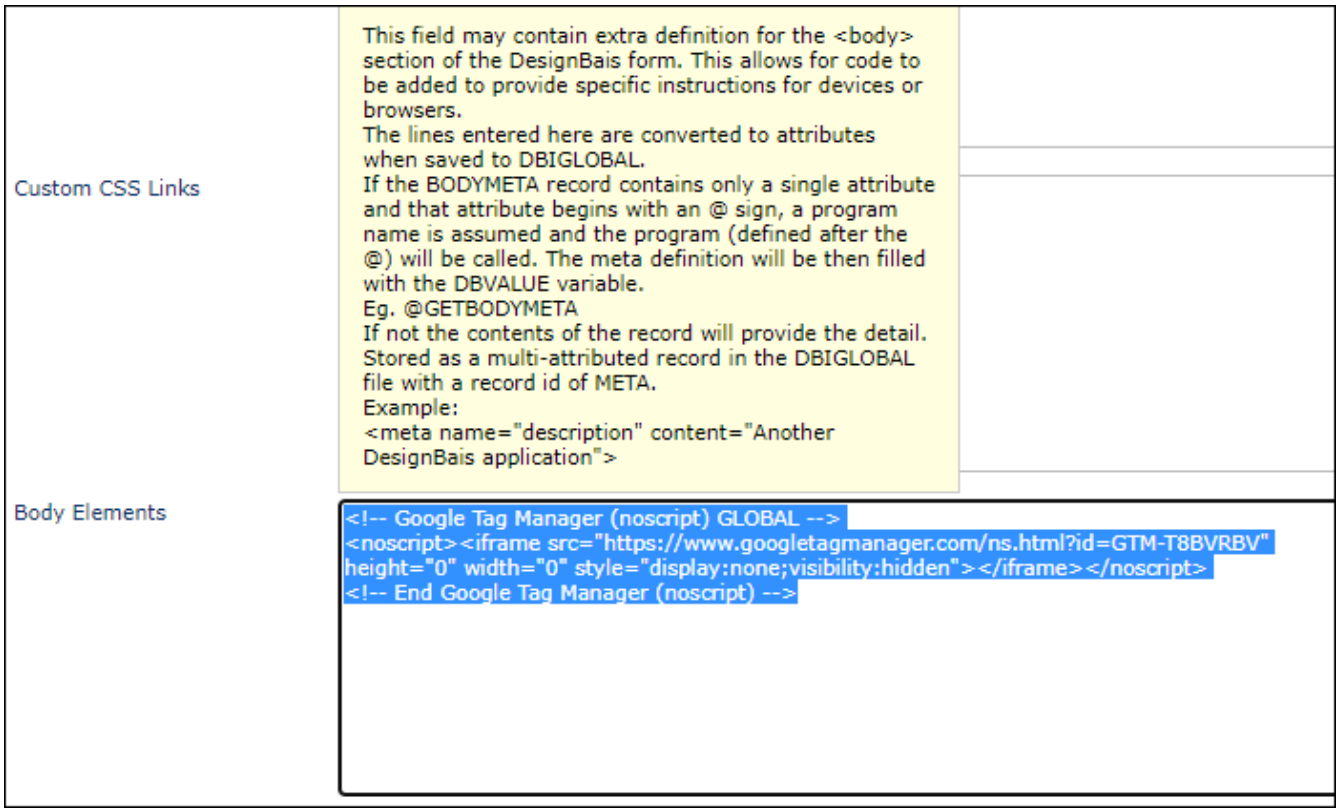
Stored as a multiattributed record in the DBIGLOBAL file with a record id of LINKS.

To explore the example shown above (<link href="myCustom.css" rel="STYLESHEET" type="text/css"/>) you can create a css file (e.g. myCustom.css) and save it in your DesignBais folder (C:\db). For an example set up a css file containing:

```
input {
   background-color:#f3f5e8;
}
```

After refreshing the browser you can see that all input boxes now have this background color unless overriden by the "style=" attributes.

Body Elements

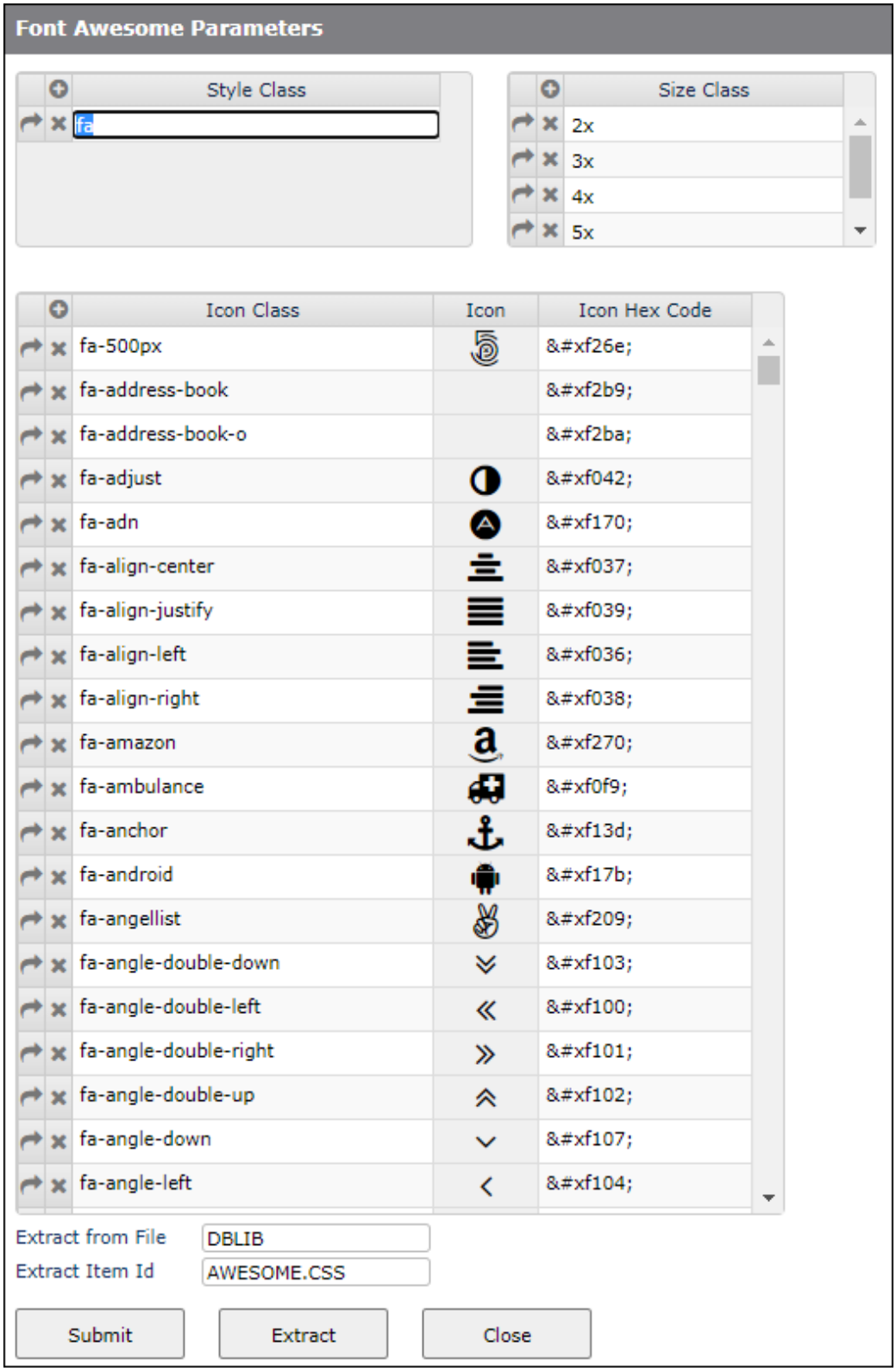


These Global Parameters may be overridden in the System Parameters General maintenance for a particular account.

Awesome Font

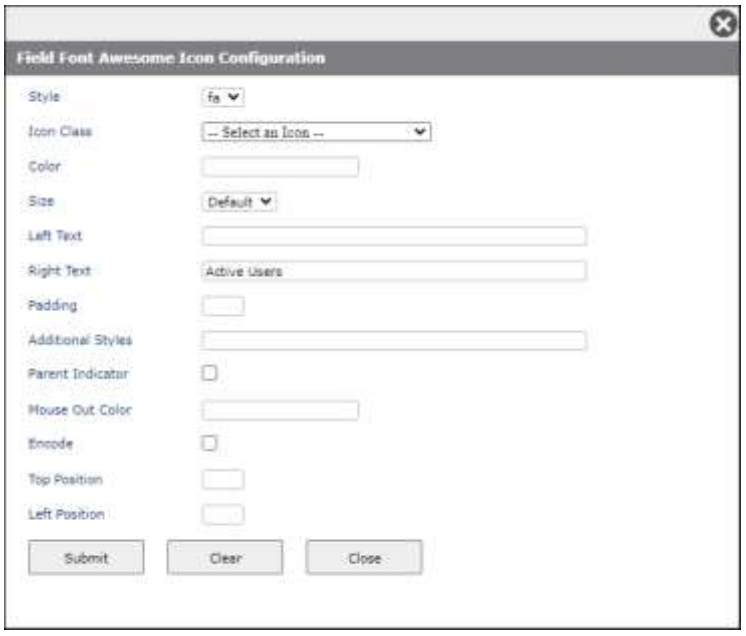
Awesome Font

Awesome Font icons can be used in form elements such as buttons, dropdown lists, grid and OFR headings. Use this form to extract the required fonts from the Awesome Font CSS files.



Awesome fonts can be applied to form elements using the awesome font settings button as shown, for example, in the Button Definition form:

Click the settings button to open the Field Font Awesome Icon Configuration form:



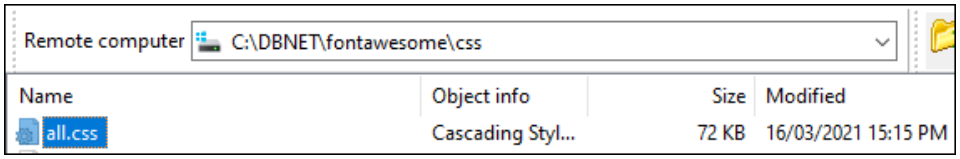
Style	Font Awesome style. The standard style (from version 4.7) was restricted to *fa*. If you have upgraded then add the extra options, such as *fab*, *far*, *fal* etc, to the Awesome Font Parameters in either System or Global Parameters.
Icon Class	Select from the list of icons that have been loaded into the Awesome Font Parameters in either System or Global Parameters.
Color	The color must be a valid HTML color suxh as *red*, *#3a3b4c* etc.
Size	Select from the dropdown list *2x* through *5x* or *lg*.
Left Text	Text that will appear on the left hand side of the icon.
Right Text	Text that will appear on the right hand side of the icon.
Padding	Font Awesome padding to appear on the left and right of the icon to separate the icon from the text
Additional Styles	Additional in-line style string to apply to the icon e.g. font-size:13pt;background-color:yellow;

Parent Indicator	This provides an @PARENT tag to the cell which will allow it to be clicked.
Mouse Out Color	Font Awesome font color to be used - must be available HTML e.g. red, #112233, etc If there is a requirement to change the default mouse-out color of the icon, then add the color to this parameter. This will be the default color for the icon. The color identified in the color parameter above will be the mouse-over color.
Encode	If the font-awesome icon and associated properties are to be displayed in a multi-value grid, then this property must be set to 1. If this is not done, the grid will fail to display.
Top Position	It is sometimes easier to position the icon within the element. This is the top position in px. E.g. 2 will position the icon 2px down from the top of the element.
Left Position	It is sometimes easier to position the icon within the element. This is the left position in px. E.g. 2 will position the icon 2px in from the left of the element.

Loading Fontawesome 5

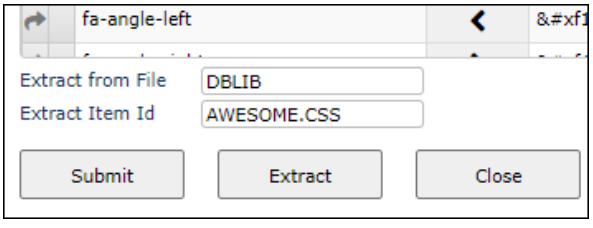
Assistance with upgrading from Version 4 can be found at this link:
https://fontawesome.com/how-to-use/on-the-web/setup/upgrading-from-version-4

DesignBais ships with Fontawesome Version 4 loaded. If you wish to load Version 5 then locate the *all.css* file in the DesignBais website, as shown below:



Copy this file into the database. This is done by copying the file into, say, a program library folder such as DBLIB. The *all.css* file contains the css for awesome fonts as shown in the copy of the first few lines of the file:

```
/*!
 * Font Awesome Free 5.15.3 by @fontawesome - https://fontawesome.com
 * License - https://fontawesome.com/license/free (Icons: CC BY 4.0, Fonts: SIL OFL 1.1, Code: MIT License)
 */
.fa,
.fas,
.far,
.fal,
.fad,
.fab {
  -moz-osx-font-smoothing: grayscale;
  -webkit-font-smoothing: antialiased;
  display: inline-block;
  font-style: normal;
  font-variant: normal;
  text-rendering: auto;
  line-height: 1; }

.fa-lg {
  font-size: 1.33333em;
  line-height: 0.75em;
  vertical-align: -.0667em; }

.fa-xs {
  font-size: .75em; }
```



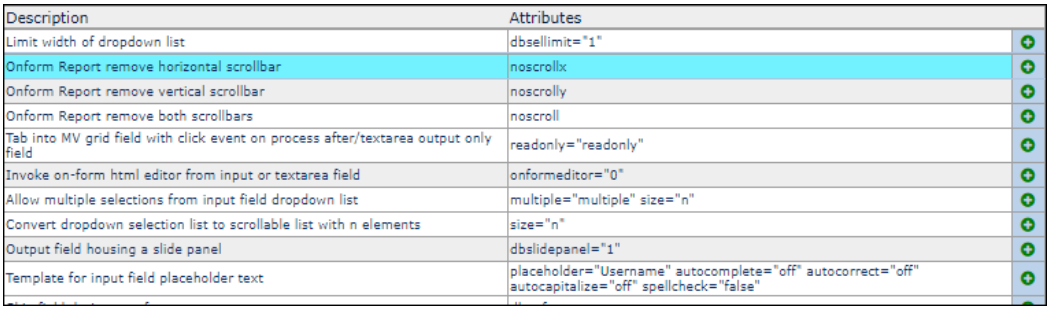
Use the extract function to load the version 5 fonts. Specify the name of the file and the item id to which you copied the *all.css* file. Then click *Extract*.

Implementing Awesome font in On-form Report.

Use the pipe-separated string shown in green to define the awesome font icon.

```
OUTPUT.AT = 'white-space:normal~|padding-right:2px~|'
ICON.ADD = '~fa|fa|fa-plus-circle|green|1x||||font-size:13px;||||||'
ZZMAX = DCOUNT(DBOTHER.RECORD(2)<1>,VM)
* ALREADY.SELECTED = DBSTORE(10)<2>
FOR ZZ = 1 TO ZZMAX
   ID = DBOTHER.RECORD(2)<2,ZZ>
   OUTPUT.LINE = DBOTHER.RECORD(2)<1,ZZ>
   OUTPUT.LINE<1,2> = ID
   OUTPUT.LINE<1,3> = ICON.ADD
   *
   OUTPUT.REPORT(PROCESS.REPORT.NUMBER)<ZZ> = OUTPUT.LINE
   OUTPUT.KEYS(PROCESS.REPORT.NUMBER)<ZZ,1> = ID
   OUTPUT.KEYS(PROCESS.REPORT.NUMBER)<ZZ,2> = ID
   OUTPUT.KEYS(PROCESS.REPORT.NUMBER)<ZZ,3> = ID
   *
   OUTPUT.ATTR(PROCESS.REPORT.NUMBER)<ZZ,1> = "alignleft~|":OUTPUT.AT
   OUTPUT.ATTR(PROCESS.REPORT.NUMBER)<ZZ,2> = "alignleft~|":OUTPUT.AT
   OUTPUT.ATTR(PROCESS.REPORT.NUMBER)<ZZ,3> = "aligncenter~|font-size:9pt~|":OUTPUT.AT
NEXT ZZ
```

The on-form report displays the awesome font in column 3.



Description	Attributes
Limit width of dropdown list	dbsellimit="1"
Onform Report remove horizontal scrollbar	noscrollx
Onform Report remove vertical scrollbar	noscrolly
Onform Report remove both scrollbars	noscroll
Tab into MV grid field with click event on process after/textarea output only field	readonly="readonly"
Invoke on-form html editor from input or textarea field	onformeditor="0"
Allow multiple selections from input field dropdown list	multiple="multiple" size="n"
Convert dropdown selection list to scrollable list with n elements	size="n"
Output field housing a slide panel	dbslidepanel="1"
Template for input field placeholder text	placeholder="Username" autocomplete="off" autocorrect="off" autocapitalize="off" spellcheck="false"

The pipe-separated string positions are as follows:

1. ~fa — Informs DesignBais that this is a font-awesome string.
2. Font Awesome Style — Determines what font-awesome style is to be used. Currently: fa, fas, far, fal, fad
3. Font Awesome Icon — The font-awesome icon from the icon-set matching the style. In the above example the fa-plus-circle is used.
4. Color — Color to be used by the icon
5. Size — This is a font-awesome size description. If blank the font-size of the container will be used. xs, sm, 2x 3x, 10x
6. Text left — Text to appear on the left of the icon.
7. Text right — Text to appear on the right of the icon.
8. Text padding — Pad the distance between icon and text by defined pixels
9. Additional Style — Any CSS elements that to be applied to the cell. In the example above the font-size of the icon is changed.
10. Parent Indicator — This provides a @PARENT tag to the cell which will allow it to be clicked if required.
11. Mouse out color

    If there is a requirement to change the default mouse-out color of the icon, then add the color to this parameter. This will be the default color for the icon. The color identified in Parameter 4, will be the mouse-overcolor.
12. Encode on/off

    If the font-awesome icon and associated properties are to be displayed in a multi-value grid, then set this property to 1. If this is not done, the grid will fail to display.
13. Position Absolute Top in px — Defines the position.
14. Position Absolute Left in px — Defines the position.
15. Additional Class — This is used to add classes to the icon such as rotate, flip, etc.
16. Title — Title attribute to be used in the HTML

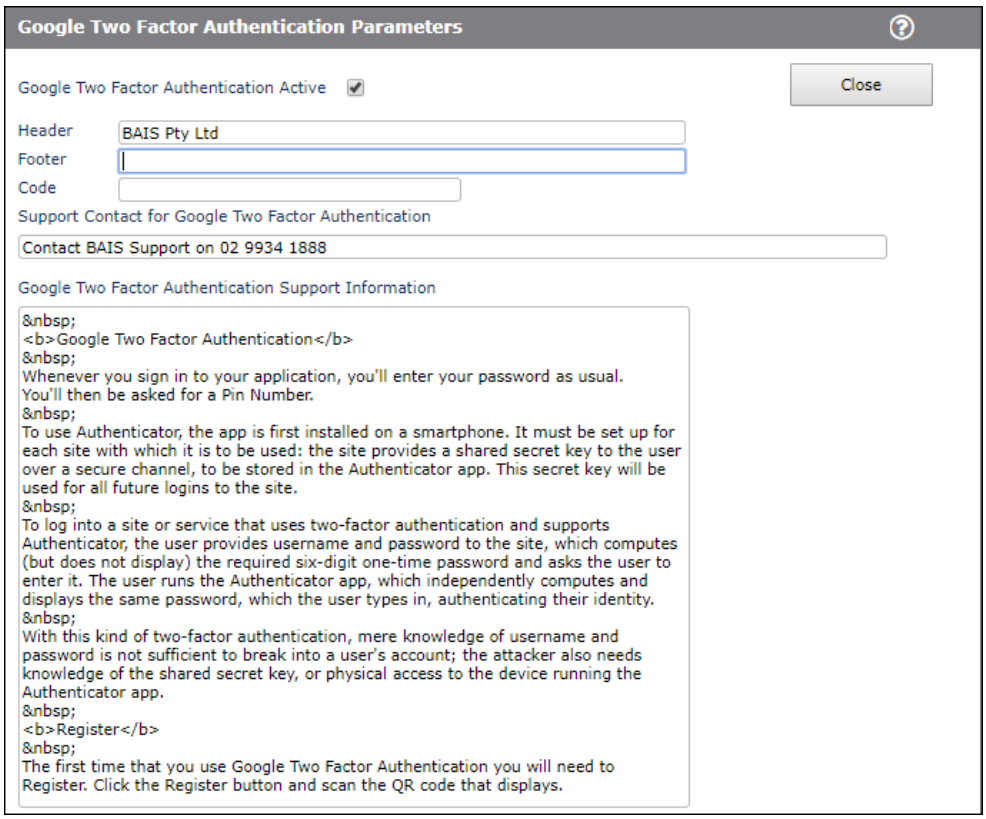
Google Authorisation

DesignBais provides the ability to implement *Two factor Authentication* using "Google Authenticator".

Users must install the Google Authenticator App on their smart phone. When first logging it is necessary to *Register*. This is only done once. At login the user must get the PIN from Google Authenticator and enter it in the text field provided.

The DesignBais System Administrator must set up the Two Factor Authentication *Header*, *Footer* and *Code* fields in the Global Login Parameters form. Note that once these three fields are entered then the *Users* maintenance form DBIUSERS_D10 will display a *PIN Required* field with the default set to *Yes*. The first user, with *PIN Required* set to *Yes*, that clicks the *Register* button on the login form, and then uses their QR Reader to register the image presented, will cause the contents of the three authentication fields to be registered with Google and made available in the Authenticator app.

If the *Code* field is altered, and a user registers these new credentials, then all users will be unable to obtain an authentication PIN until they re-Register their phone by displaying and scanning the (new) QR image.



Google Two Factor Authentication Active
: Flag to indicate that Google Two Pass Authentication is active globally. Authentication can be set to not active in a particular account by setting the equivalent flag in System Parameters.

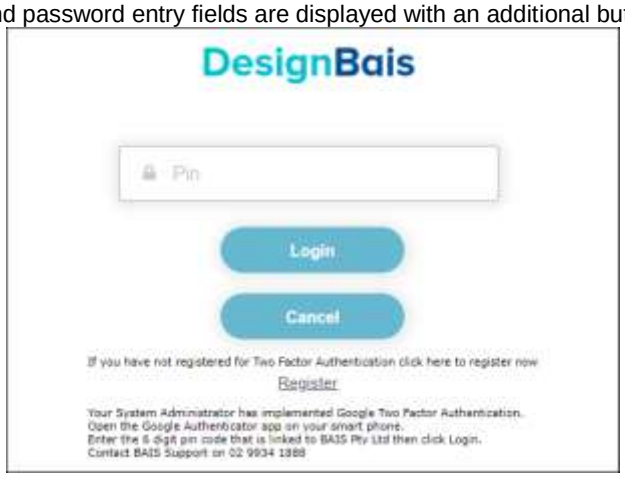
Header
: The QR Barcode Header text for Google Two Pass Authentication.

Footer
: The QR Barcode Footer text for Google Two Pass Authentication which must no contain spaces.

Code
: The pass code for Google Two Pass Authentication - 8 to 32 characters.

The following example demonstrates the procedure.

Login and password entry fields are displayed with an additional button *Register*.



If *Register* is clicked then the QR phone validation image is displayed.

The user logs in with valid credentials and clicks *OK*.

The additional Pin entry field displays and the user must enter the pin displayed by Google Authenticator on their phone.

To implement Two Factor Authentication in a login form use the new DesignBais common variable *DBQRAUTH* which is multivalued. There are two new events associated with Two Factor Authentication named *REGISTER* and *VALIDATEPIN*.

To trigger the registration process set the code to “R”. This will invoke the DesignBais event *REGISTER*.

DBQRAUTH<1,1> = "R”	PROCESS.PARAMETER returns the Registration QR Image
DBQRAUTH<1,2> = "myeventsource	PROCESS.EVENTSOURCE will be set to the value *myeventsource*
DBQRAUTH<1,3> = "mysubroutine"	The name of the subroutine for processing the registration event

After the user enters a pin number set the code to “V” to trigger the *VALIDATEPIN* event.

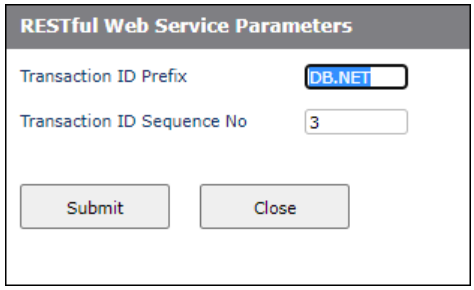
DBQRAUTH<1,1> = "V"	VALIDATEPIN will return a OK or FAILED in PROCESS.PARAMETER
DBQRAUTH<1,2> = "myeventsource	PROCESS.EVENTSOURCE will be set to the value *myeventsource*
DBQRAUTH<1,3> = "mysubroutine"	The name of the subroutine for processing the pin validation event
DBQRAUTH<1,4> = “PIN number”	The value entered in the Pin field to be validated

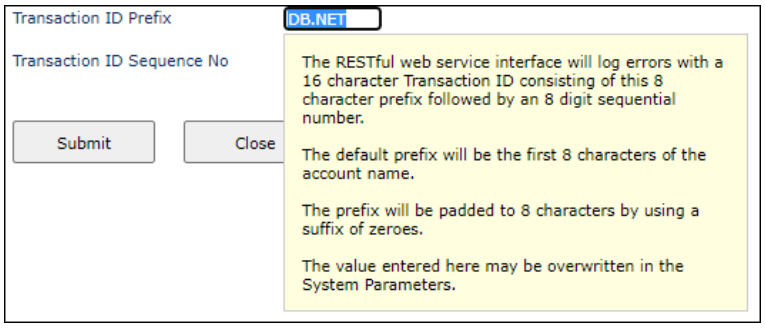
Example code:

```
CASE THIS.EVENT = "REGISTER"
        GOSUB REGISTER
CASE THIS.EVENT = "VALIDATEPIN"
        GOSUB VALIDATEPIN

CASE EVENTSOURCE = "B.REGISTER"
        DBQRAUTH      = "R"  ;* REGISTER will return a QR Image to be displayed in PROCESS.PARAMETER
        DBQRAUTH<1,2> = "REG1"                ;* Will be returned as PROCESS.EVENTSOURCE
        DBQRAUTH<1,3> = "DBI.I.DBIGLOBAL"     ;* Subroutine for return processing



REGISTER:
   * Google Two Pass Authentication
   * REGISTER = QR Barcode Image returned in PROCESS.PARAMETER
   BEGIN CASE
      CASE EVENTSOURCE="REG1"
         * Replace/Display Image
         DBIMAGESPEC<1,-1> = "db\white.gif"
         DBIMAGESPEC<2,-1> = THIS.PARAMETER
         TITLE.TXT = "Scan with the Google Authentication App to register your phone"
         CALL DBI.G.GLOSSARY(TITLE.TXT)
         DBIMAGESPEC<3,-1> = TITLE.TXT
         ALT.TXT = "Image Unavailable"
         CALL DBI.G.GLOSSARY(ALT.TXT)
         DBIMAGESPEC<4,-1> = ALT.TXT
         * Display Section containing the pin entry field
         DBWORK<DBIGO.GA.DISPLAY.WK>=2
   END CASE
   RETURN

VALIDATION:
   BEGIN CASE
      CASE FNAME="DBIGO.GA.PIN.WK"
         IF DBVALUE#'' THEN
            * Entered PIN needs validation
            * PROCESS.EVENT will be returned as VALIDATEPIN
            * OK or FAILED will be returned in PROCESS.PARAMETER
            *
            DBQRAUTH      = "V"      ;* VALIDATEPIN will return a OK or FAILED in PROCESS.PARAMETER
            DBQRAUTH<1,2> = "PIN1"                ;* Will be returned as PROCESS.EVENTSOURCE
            DBQRAUTH<1,3> = "DBI.I.DBIGLOBAL"     ;* Subroutine for return processing
            DBQRAUTH<1,4> = DBVALUE               ;* PIN to validate
         END

VALIDATEPIN:
   * Google Two Pass Authentication
   * VALIDATEPIN = OK or FAILED returned in PROCESS.PARAMETER
   BEGIN CASE
      CASE EVENTSOURCE="PIN1"
         * Recheck User, Password & PIN
```

```
      IF THIS.PARAMETER="OK" THEN
         * Validate Password
         GOSUB VALIDATE.USER
         IF IERR.TEXT#'' THEN RETURN
         GOSUB LOGIN.USER
      END ELSE
         IERR.TEXT = 'Invalid PIN, please try again ':THIS.PARAMETER
         CALL DBI.G.GLOSSARY(IERR.TEXT)
      END
END CASE
RETURN
```

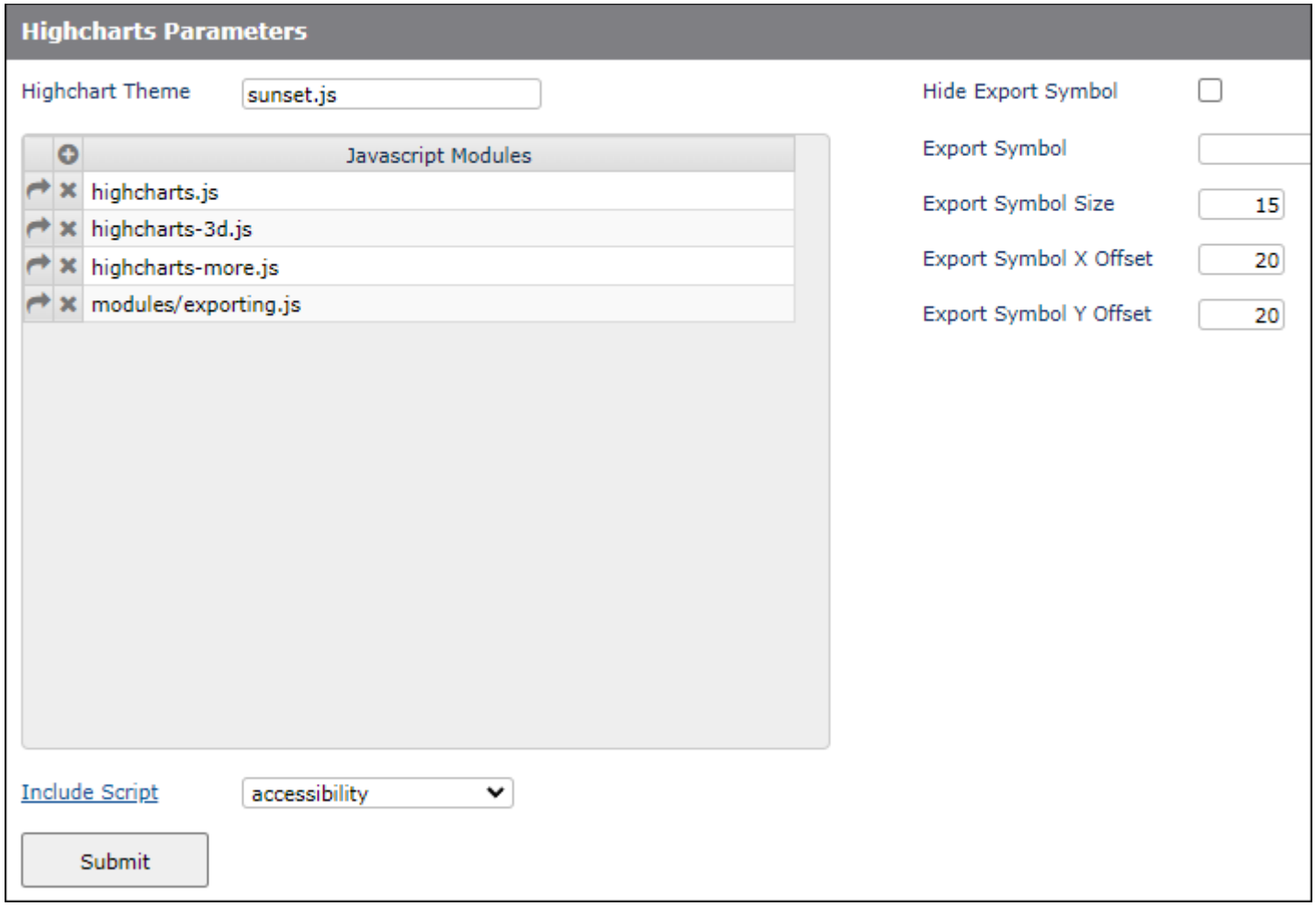
RESTful Web Service





Highcharts

Use this form to set defaults for Highcharts. The Highchart Theme can also be set via the System Parameters, so that it can vary across accounts.



Highchart Theme

The name of the Highchart Theme to be used as a default for all applications. The name entered here will be overridden by any application Hichart Theme System Parameter. If both are left blank then the default Highchart Theme is applied. Example: "dark-green.js

The available theme javascript files are held in the DesignBais website folder charts/themes. This is a javascript file which is loaded when the url is refreshed or on change of account.

<script src="charts/themes/brand-dark.js?v=21" type="text/javascript"></script>

The string “charts/themes/” is pre-pended to the name provided as per:

```
IF HICHARTS.THEMES.ON THEN
   OUTRECORD<-1> = '<script src="charts/themes/':HICHART.THEME:'?v=':WEBVERSION:'"
   type="text/javascript"></script>':HLF
END
```

If the name does not end with “.js” then the suffix is added.
Stored in the DBIGLOBAL record HICHART.

Javascript Modules The list of Highchart modules to be included as a default for all applications. The list entered here will be overridden by any System Parameters list. If Global and System lists are left blank then the default Highcharts modules are applied. These are charts/hicharts.js, charts/hicharts-3d.js and charts/hicharts-more.js
The available javascript files are held in the DesignBais website folder charts and charts/modules.

Include Script The dropdown list of available Highchart javascript modules. Click the hyperlink to add the selected module to the list of Javascript Modules above.

Hide Export Symbol The export image may be hidden. The image parameter entered here will be overridden by the System Parameter value. If both are left blank then the default Highchart image is applied.

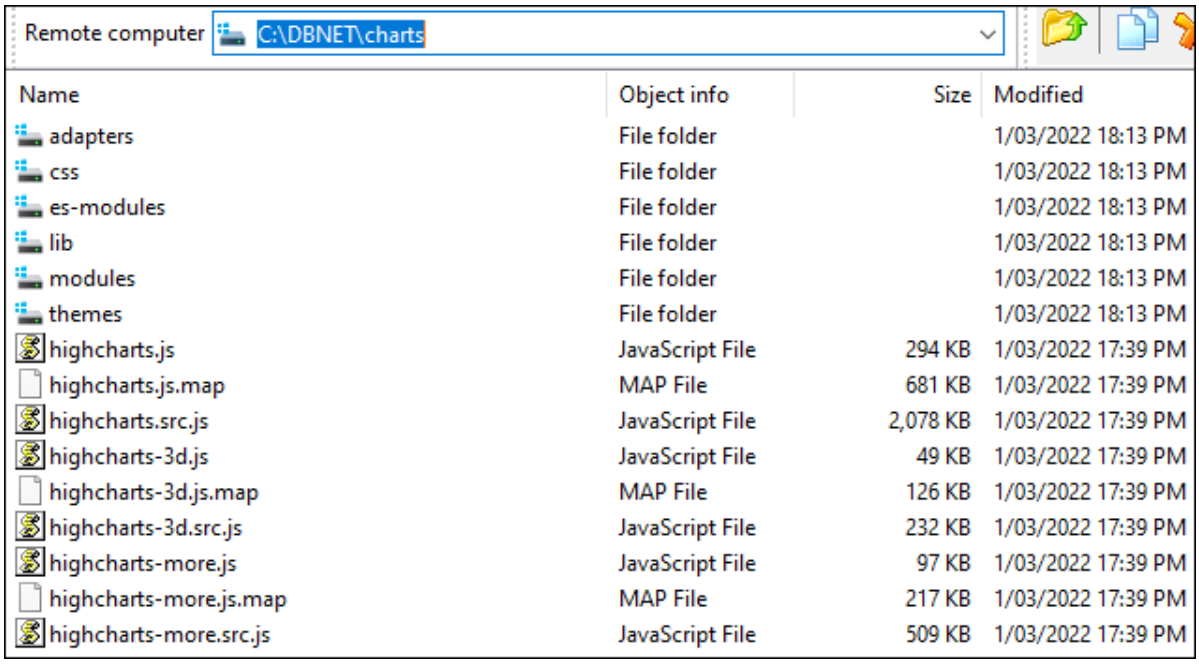
Export Symbol The image path of the image to replace the Highchart default. The image entered here will be overridden by the System Parameter value. If both are left blank then the default Highchart image is applied.

Export Symbol Size The size of the export image symbol to be used as a default for all applications. The size entered here will be overridden by any System Parameter value.

Export Symbol X Offset The X offset of the Export Image Symbol to be used as a default for all applications. The offset entered here will be overridden by the System Parameter value. If both are left blank then the default Highchart position is applied.

Export Symbol Y Offset The Y offset of the Highchart Export menu image symbol to be used as a default for all applications. The value entered here will be overridden by any System Parameter entry. If both are left blank then the default Highchart position is applied.

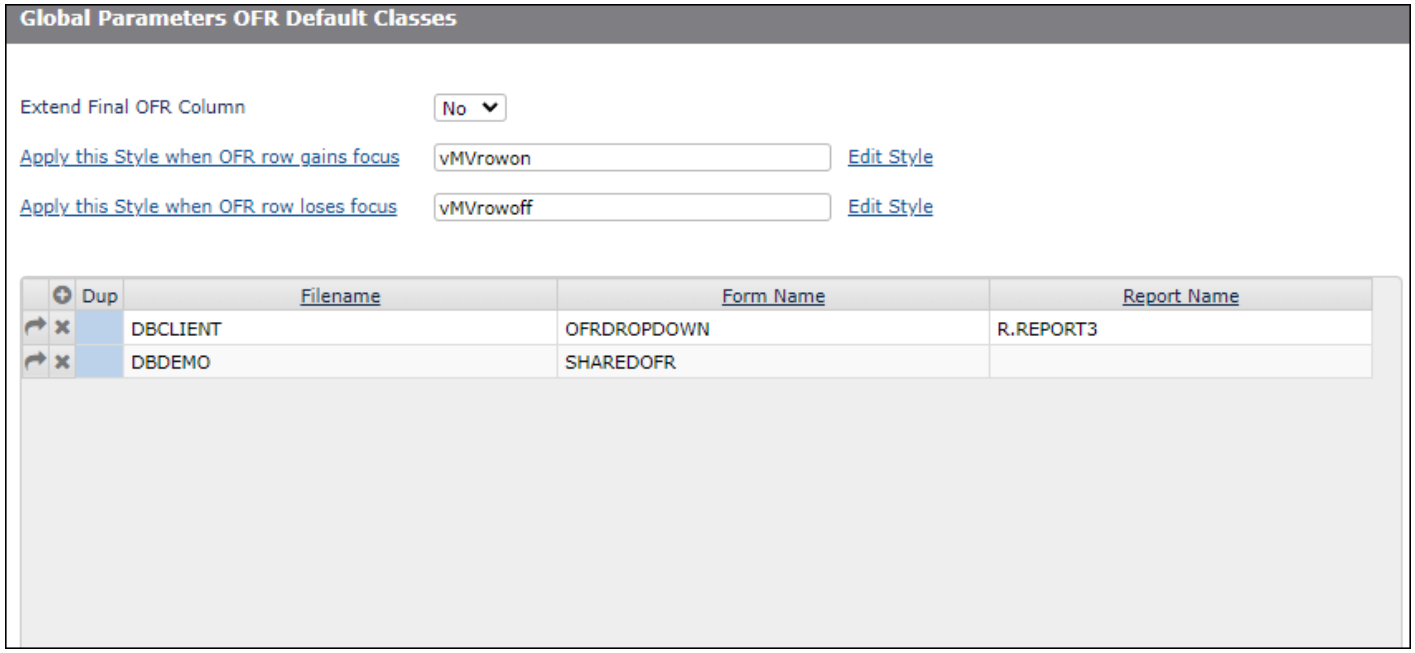
For the website DBNET:



OFR Default Classes

Use this form to set the flag that controls the determination of on-form report column width.

Use this form to designate two styles to be used when on-form report rows gain and lose focus and to maintain a list of forms to which these styles are to be applied. These parameters are held on DBIGLOBAL record OFR.DEFAULTS.



Extend Final OFR Column

Set this flag to cause the final column of on-form reports to extend into the space that would be occupied by a vertical scroll bar.

When set, and a vertical scroll bar is required, then the scroll bar will display over the right hand side of the last column.

This flag effectively allows the columns of the report to occupy the entire width of the report container.

Apply this Style when OFR row gains focus

Specify a style to be applied to OFR row on gaining focus.

Apply this Style when OFR row loses focus

Specify a style to be applied to OFR row on losing focus.

Filename

The filename of a form for which specified report fields on the form are to use the specified styles for gain and loss of focus.

Form Name

The form name for which specified report fields on the form are to use the specified styles for gain and loss of focus.

Report Name The name of a report field on the form that is to use the specified styles for gain and loss of focus. Leave blank if all on-form reports on the form are to use the specified styles.

Edit Style button Click to review or maintain the two styles entered in the associated field to the left.

Chapter 26 – Session Reinstatement

Session Reinstatement

DesignBais allows the developer to link to an external website and return to the current form. This is achieved by creating a unique return key which is saved and sent to the browser. On return from the external website the browser passes this key back, thus allowing DesignBais to reinstate the DesignBais session that initiated the external call.

The following example demonstrates this function.

The trigger to cause DesignBais to reinstate a session is to populate attribute 1 of the common variable DBPARKSESSION with the name of the basic routine that will control the processing of the call.

The originating form could have a button to initiate a call to an external website using the DesignBais common variable DBCALLURL

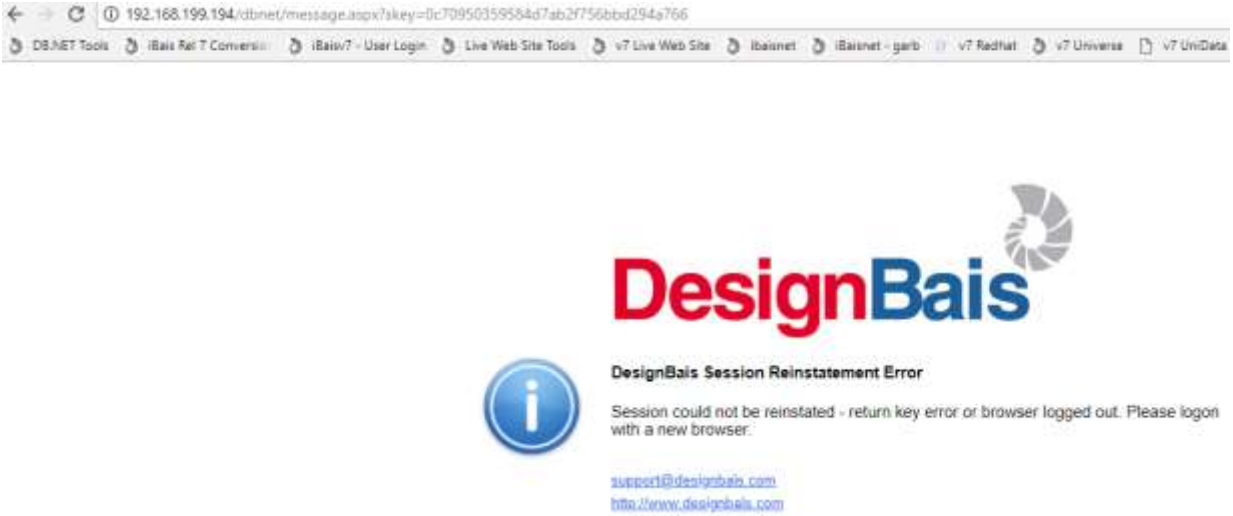
```
CASE EVENTSOURCE = "B.DBRIS" AND SCREEN.NO = "DBRIS"
   DBCALLURL = 'http://192.168.199.25/bc/v7x.asp'
   DBPARKSESSION = "DBC.I.DBCLIENT"
```

The developer must tell the external website where to return to. This is done using DBW3CQSTRING as shown below. Additional information can be returned via this query string in the normal manner.

Your basic routine can then utilise the two events within DesignBais called REINSTATE.SESSION and REINSTATE.SESSION.POST. Use the REINSTATE.SESSION.POST to perform updates to DBRECORD etc.

DBCALLURL	Contains the URL to the destination site. For example: DBCALLURL = 'http://192.168.199.25/bc/v7x.asp'
DBPARKSESSION	Contains the name of the subroutine to be called on return from the external website.
DBW3CQSTRING	On reinstatement contains the string "dbretkey=AAAAANNNNN" where AAAAANNNNN is the unique key used to designate the DesignBais session that initiated the call to the external website. The developer can append other values to this query string in the normal way: http://192.168.199.194/db7009/?dbretkey=QRCHJ16944&fred=abc&george=cde

If the user logs out of DesignBais before the session re-instatement is completed then the following error may display. This reflects the fact that the browser COOKIE is not left open after a log out. It is done this way so that the user only has to logout of one tab. Other tabs can be closed by the "X". DesignBais does a fresh count of user connections if more than 20 minute elapses between hits. If there is a hit in a browser tab that has been idle for over 20 minutes then the browser cookie is re-created.



Decoding a URL Query String

The URL query string is stored in the DesignBais common variable DBW3CQSTRING.

URL encoding is necessary to convert the query string into a format that can be transmitted over the internet since the internet requires characters from the ASCII character set. Any character that is outside the ASCII set must be encoded before being transmitted over the internet.

A URL cannot contain a space character. URL encoding will replace a space character with a “plus” (+) sign or with the hexadecimal %20.

Other non-ASCII characters will be encoded as percentage (%) followed by 2 hexadecimal digits.

The structure of the query string containing key-value pairs is typically:

x=1&y=2&z=3

In the above example the key-value separator is the “equals” (=) character.
The key-value delimiter is the “ampersand” (&) character.

When unencoding a query string the first step is to reinstate the space characters and to isolate the key-value pairs in the query string.

Then you can use a call to the DesignBais subroutine DBI.G.ENCODENET(DBW3CQSTRING,PASSVAL)

- Set PASSVAL = "ESCAPE" to encode a URL query string
- Set PASSVAL = "UNESCAPE" to unencode the query string

In the following example the “UNESCAPE” option is used to convert the “%nn” hexadecimal values back to ASCII characters.

```
IF DATA.STR # "" THEN
  DBW3CQSTRING = DATA.STR
  DBW3CQSTRING = CHANGE(DBW3CQSTRING,"%26ajx%3Dyes","")
  * Treat initial %2B (+) as a space
  DBW3CQSTRING = CHANGE(DBW3CQSTRING,"%2B"," ")
  CALL DBI.G.ENCODENET(DBW3CQSTRING,'UNESCAPE')
  * Convert dbpage value to lowercase
  PG.IDX = INDEX(DBW3CQSTRING,"dbpage=",1)
  IF PG.IDX THEN
     START.QS = DBW3CQSTRING[1,PG.IDX+6]
     DBPAGE = FIELD(DBW3CQSTRING[PG.IDX+7,LEN(DBW3CQSTRING)-PG.IDX-6],"&",1)
     END.QS = DBW3CQSTRING[PG.IDX+7+LEN(DBPAGE),LEN(DBW3CQSTRING)-PG.IDX-LEN(DBPAGE)-6]
     CALL DBI.G.ENCODENET(DBPAGE,'UNESCAPE')
     DBW3CQSTRING = START.QS:OCONV(DBPAGE,"MCL"):END.QS
  END
END
```

Steps to follow to decode a query string

qstring ="aaaa%3Dbbbb%2Bcccc%2Bdddd%26ffff%3Dgg%252bgg%2526hh"

hex values are case insensitive: %2b = %2B
"+" and SPACE arrive as %2B. They mean the same thing [SPACE] in a query string. If the user really means "+" then s/he must enter %2B which arrives as %252b.

STEP 1 Decode all percents [%nn] (* see below - Percent decoding of reserved characters)**:**
In this example:
%3d = "="
%2b = "+"
%25 = "%"
%26 = "&"

aaaa=bbbb+cccc+dddd&ffff=gg%2bgg%26hh

STEP 2 Convert all + signs to spaces

aaaa=bbbb cccc dddd&ffff=gg%2bgg%26hh

STEP 3 Parse values (using & and = signs)

aaaa=bbbb cccc dddd
ffff=gg%2bgg%26hh

STEP 4 Decode all percents again

aaaa=bbbb cccc dddd
ffff=gg+gg&hh

(*) Percent decoding of reserved characters:

%20	SPACE	%2c	,	%5c	\
%21	!	%2d	-	%5d	]
%22	"	%2e	.	%5e	^
%23	#	%2f	/	%60	`
%24	$	%3a	:	%7b	{
%25	%	%3b	;	%7c	|
%26	&	%3c	<	%7d	}
%27	'	%3d	=	%7e	~
%28	(	%3e	>		
%29	)	%3f	?		
%2A	*	%40	@		
%2B	+	%5b	[		

Chapter 27 – Code Editor

Code Editor

The DesignBais Code Editor is based on the Ace Editor. Ace is an embeddable code editor written in JavaScript.

You will welcome the improvements that come with this editor such as retaining the last file and record name, improved Find (Ctrl F).

On first opening Code Editor where Anonymous authentication is used the user will be prompted for a Windows username and password. Basic or Windows authentication do not require a login prompt on code editor because the user is authenticated already on entry to DesignBais.

Note that a user must have the Code Editor Access field set to true in order to access the Code Editor. Refer to the User Maintenance chapter. If a user is not flagged for access then the following message appears:

Code Editor access requires:

- Allow access for the user by setting the Code Editor Access flag to Yes in DesignBais Users maintenance
- Browser popups must be allowed
- When credentials are requested at initial login of the Code Editor they must be for a Windows user on the Web server
- This Windows user must be listed in the web.config code editor section
- Basic authentication must be enabled.

Records being edited have exclusive locks set. In the event of the editor reporting that a record is locked when it is not being edited remember to check and release any exclusive lock that may still be present. The editor will display a message like this:

```
Unable to gain an exclusive read on the required record. The record is currently locked with exclusive access by garb at 10:54:24 on 02 MAY 2017
```

Code editor creates an additional lock in DBISESSIONS with id of EDITOR*checksum which may also have to be removed. For example: "EDITOR*2456" where 2456 is the checksum of the path to the locked record id.
The full path to the record is held in attribute 6. All attributes are multivalued.

<1>: 19857 The date the lock was set
<2>: 66928.797 The time the lock was set
<3>: legj The user record that holds the lock
<4>: DB.NET The name of the account that the locked record is in
<5>: 1 The number of locked records held by this checksum
<6>: E:\HOME\DESIGNBAIS\DBINET\DBINET*DBI.G.RESETLNET

Code editor locks are held in DBISESSIONS. If database accounts do not share the DBISESSIONS file then the locks will not prevent concurrent access to the same record. System Administrators must take this into account when setting up DesignBais enviroments. If distinct database accounts can lock the same items then these accounts need to share session files.

This situation can arise very simply where a common program library is utilised by two database accounts. DesignBais will set locks when Code Editor loads a record for editing. These locks will be identical but if the accounts have different sessions files then DesignBais cannot know about a lock in the other account.

The DesignBais Tools Editor Button

The Editor button on the DesignBais Tools form opens a form which displays a list of the 50 most recent items that have been opened in the Code Editor by this user.

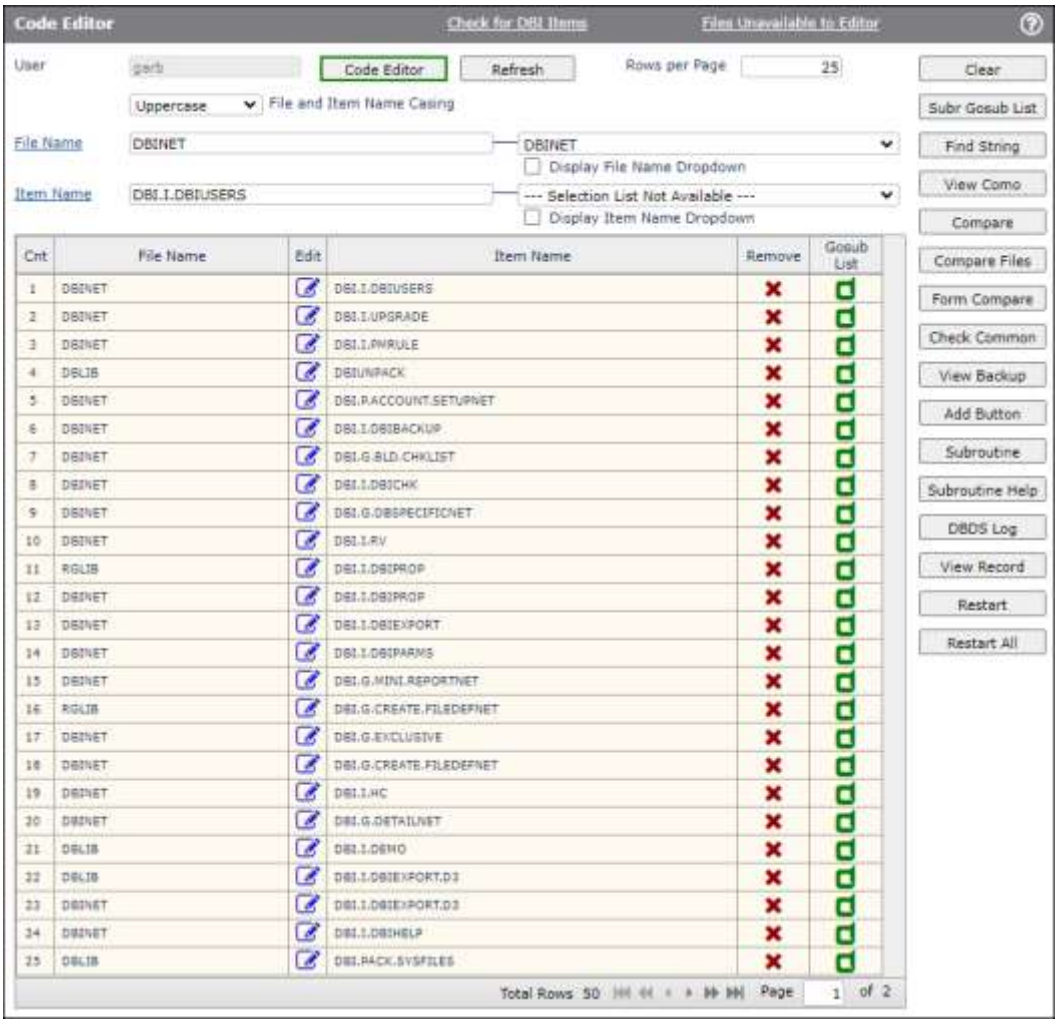
From this form click the ***Code Editor*** button to run the Code Editor as normal. You may need to enable pop-ups the first time.

- Alternatively select an entry from the list of previously edited items.
- Click the required row in the Edit column to edit the item on that row.
- Click the File Name column to load the file name in the clicked row into the File Name field.
- Click the Item Name column to load the item name in the clicked row into the Item Name field.
- Click the Code Editor button to invoke the Code Editor. The values in File and Item Name will be passed in.

When a File Name is entered the dropdown selection list for Item Name is populated. Selecting from this dropdown will load the selected item into the Item Name field. Click the Code Editor button to edit the selected item. Note that you cannot use file names containing '&' such as the '&PH&' file in Universe. In this case set up a q-pointer name that does not contain '&'.

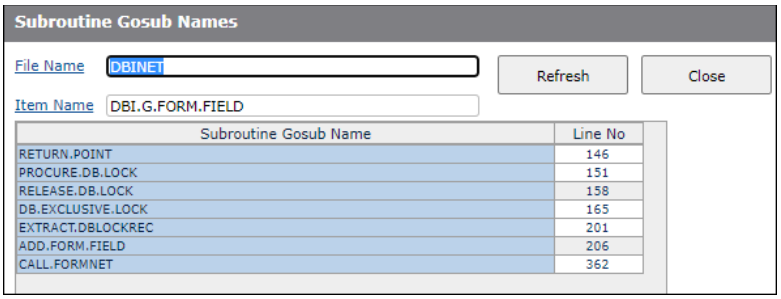
Click the Remove column to remove the item in the clicked row from the list.

Items passed to the Code Editor in the above manner will be added to the list of previously edited items. Note that items called directly from within the Code Editor will only be added if the ***Refresh*** button is clicked while the item is still open within the editor.



Alternatively click the green list icon in column 6 "Gosub List" to display the list of Gosub Names in the selected routine.

You can then click on any Gosub Name and the code editor will open the subroutine with focus on the selected line number.



Refer to the screenshots above:

The ***Find String*** button accesses the Review Basic Library option available on the Upgrade Details menu. This form allows the developer to find selected strings in any basic library file, or indeed any normal file, to edit the items containing the strings, to apply a replacement string, and to then compile each routine (in the case of the searched file being flagged as a basic library file). Refer to the notes at the end of this section.

The ***Compare*** button opens the compare option, allowing the developer to compare 2 items. Refer to the notes at the end of this section.

The ***Compare Files*** button opens a tool to allow you to compare multiple files across two accounts.

The ***Form Compare*** button opens the compare option, allowing the developer to compare 2 form records.

The ***DBDS Log*** button opens a form that displays the accumulated output from the DBDS DesignBais common variable. Useful for viewing DBDS output after clicking OK to the display in the Alert box, or to see DBDS output that may have been truncated by the Alert box. See *DBDS Log* section in this Manual.

The ***View Como*** button displays the contents of the como for the currently logged in user id. On the View Como form there is a button *Phantom Output* which displays selected records generated by phantom processes.

The ***View Backup*** button opens the Backup form. See the *Backup Changed Items* section in this Manual.

The ***Add Button*** allows the developer to add to and remove from a form the Form Help and the Display Common Variables buttons. The Form Help button function is described in the *Form Help* section of this manual. The Display Common Variables function is described in the *Display Common Variables* section of this manual.

The ***Subroutine*** button accesses the *Create New Subroutine and Insert Basic Code* Segment form (DBIUSERS_D85). The function of this form is described in the *Code Editor* section of this manual.

The ***Subroutine Help*** allows developers to provide documentation for standard subroutines used throughout the application. Designbais subroutines are also documented here, as well as in the DesignBais Reference Manual.

The ***Check Common*** button allows developers to view the list of DesignBais common variables and to check is a variable name that a developer plans to use is in fact the name of a common variable.

The ***View Record*** button opens a form that permits the developer to view any record on any file. It is particularly useful for viewing multivalued, associated attributes on a record, allowing the developer to click through each value progressively.

The ***Restart*** and ***Restart All*** buttons allow developers to drop all IIS connections in order to permit new object code to take effect. See the comments in the *Basic Restart* option under *System Parameters* in this manual.

The ***Command Line*** option provides a means of executing a limited set of TCL commands from within DesignBais. This is useful if a telnet session is not available.

The ***Subr Doco*** button and the four buttons below it are for DesignBais use only.

The Code Editor

The top menu has the following options:

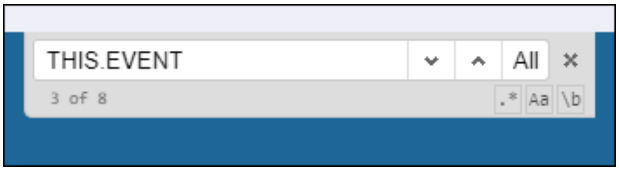
- FILE
- EDIT
- THEMES
- FONT SIZE
- FONT FAMILY
- HELP

These options are self-explanatory by clicking and reviewing the available entries.

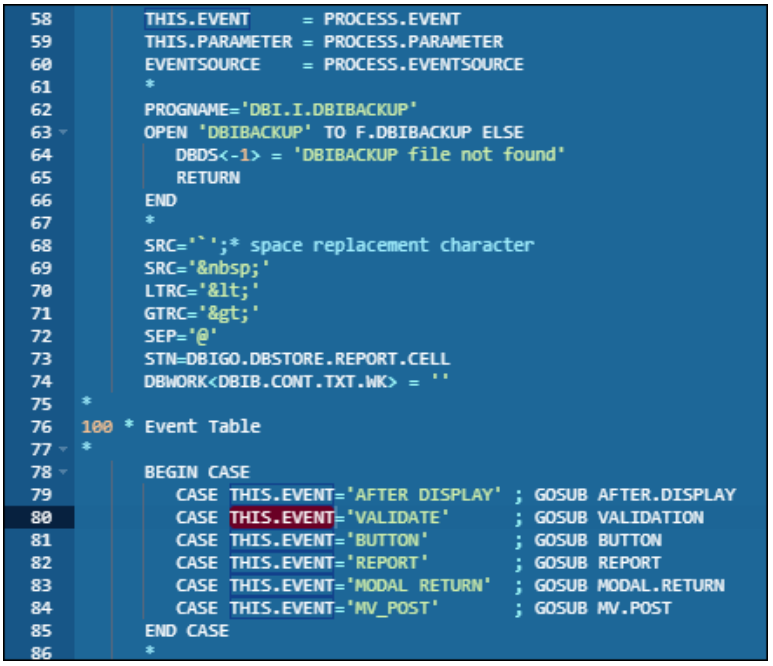
Code Editor Edit Menu – Find option

The EDIT menu “Find” entry displays a window in the top right corner of the editor pane. This can be activated by entering CTRL+F. After entry of the search text use the up and down arrows to display each occurrence. As long as focus remains on the search box you can also use the ‘Enter’ key to move to each occurrence.

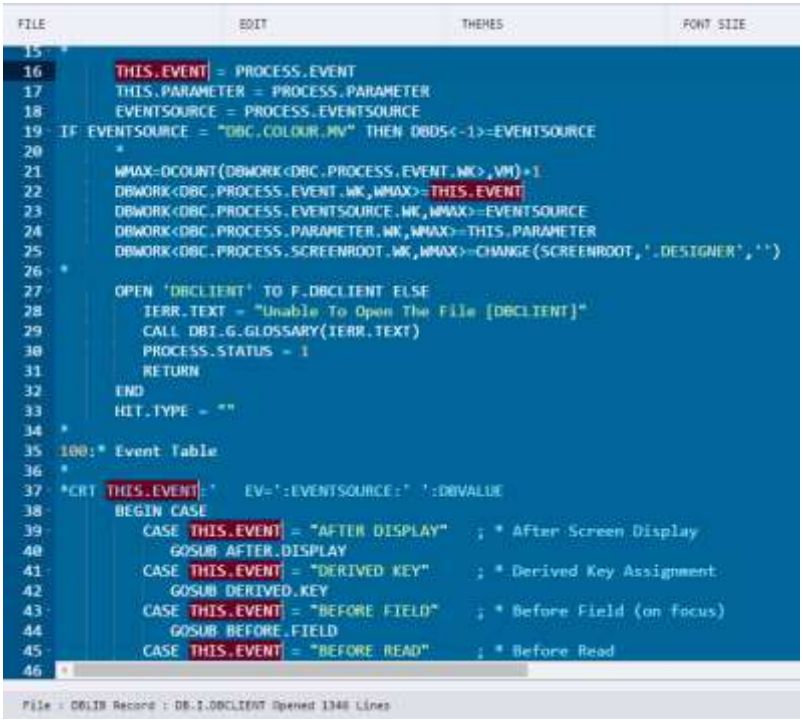
Pressing the ‘Enter’ key also triggers the display of ‘nn of nnn’ showing the number of occurrences of the string.



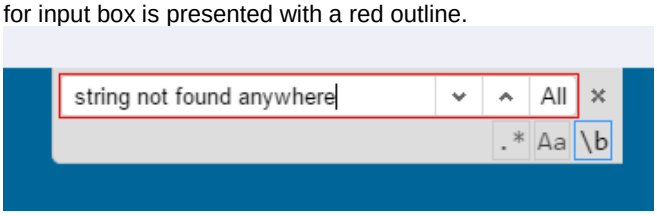
Note that there is a faint border around all occurrences of the search string (blue border) while the current occurrence is highlighted in red.



Or click “All” to see all occurrences highlighted in red within the editor pane.



If the search string is not found in the record then the Search for input box is presented with a red outline.



The “.*” button for a RegExp search is not implemented.

Click the “Aa” button to switch between case sensitive and insensitive search mode.

Click the “\b” button to perform a whole word search.

Each of these 3 buttons is outlined in blue if the button is active.

Note that to use the Find Next (CTRL+K) shortcut you must first click on the editor pane. If focus is still on the Find box pressing the enter key will skip through each occurrence of the search string.

Code Editor CTRL+G or F3 option

There is a CTRL+G (or F3) option that is useful in conjunction with the Find option. Enter CTRL+G after loading a record into the editor. A display box will open at the top right. Enter a string that you want to locate. All occurrences of the string will be highlighted on the currently displayed page and you can arrow up or down through them. Note that this action does not scroll the editor pane whereas the arrow up and down on the Find function will scroll through all occurrences within the record.

So it is useful to enter a different string in the CTRL+G box and use the CTRL+F to locate one string, then use the arrow navigation buttons on the CTRL+G box.

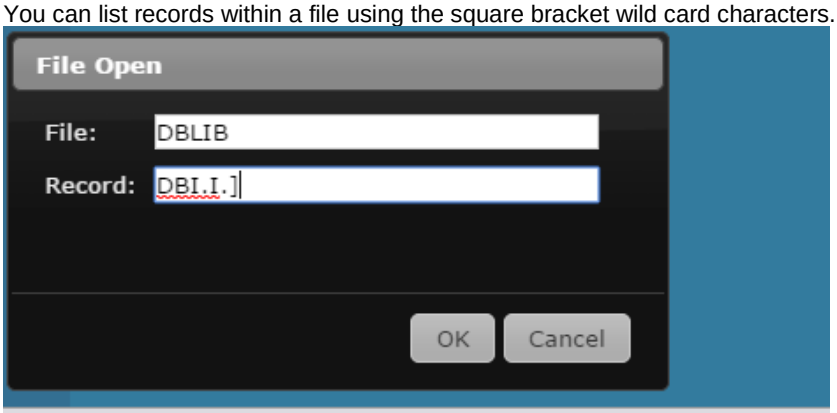
The value entered in the CTRL+G box is retained when saving the record being edited.

In the example below the CTRL+F string is “PASSENQ” which is highlighted in red on the pane. The CTRL+G string is “record” with occurrences highlighted in orange and yellow.

```
408             GOSUB MAINTAIN.RECORD
409
410          CASE EVENTSOURCE='PMSTP.GROUP'
411             IF DBOTHER.RECORD(19)<DBMVCOUNT,PMSTP.GROUP> # '' THEN
412                MAN.BY.GRP = DBOTHER.RECORD(19)<DBMVCOUNT,PMSTP.GROUP>
413                GOSUB CHECK.WF.ACCESS
414                IF WF.ACCESS.ALLOWED THEN
415                   DBSTORE(DBIGO.DBSTORE.GEN)<1>=DBOTHER.RECORD(19)<DBMVCOUNT,PMSTP.GROUP>
416                   DBPASS.DBVALUE=DBOTHER.RECORD(19)<DBMVCOUNT,PMSTP.GROUP>:VM:DBWORK<PMSTS.PROC.WK>:VM:'R'
417                   DBPASS.DBVALUE.TO='PMSTS.GROUP.WK':VM:'PMSTS.PROC.WK':VM:'PMSTS.DISPLAY.WK'
418                   PROCESS.STACK='DBIPMSTATUS_GROUPENQ~L'
419                END
420                PROCESS.PARAMETER = SAVE.PROCESS.PARAMETER
421             END
422
423          CASE EVENTSOURCE='PMSTS.REMAIN.STEP'
424             * does this code get executed ?
425             * GOSUB DISPLAY.STEP.DETAILS
426
427          CASE EVENTSOURCE = 'B.MAINTAIN'
428             STEPID=DBRECORD<PMSTS.REMAIN.STEP,1>
429             IF STEPID = '' THEN STEPID = DBRECORD<PMSTS.CURR.STEP>
430             IF STEPID = '' THEN STEPID = DBRECORD<PMSTS.LST.CMP.STEP>
431             RECORDID = DBWORK<PMSTS.RECORD.ID.WK>
432             GOSUB MAINTAIN.RECORD
433
434          CASE EVENTSOURCE = 'B.PASS'
435             PROCESS.STACK = 'DBIPMSTATUS_PASSENQ~L'
436
437       END CASE
438
```

Using Wild Card Characters

You can list records within a file using the square bracket wild card characters.




```
SORT DBLIB WITH @ID = "DBI.I.]" NOPAGE 01:02:51pm 02 May 2017 PAGE 1
DBLIB.....................................

DBI.I.COMPFORM.BU
DBI.I.DBIPARMS.BACKUP2017.03.03
DBI.I.DBIPARMS2017.03.04
DBI.I.DEMO
DBI.I.DEMO.OFR

5 records listed.
```

Calling the Code Editor from a Basic Subroutine

If you want to call the Code Editor from within a basic subroutine use DBCALLURL.

In the example below the Code Editor is called from a cell in column 2 of an On-form Report. OUTPUT.KEYS is set to contain the File Name and Record Id separated by a pipe character.

```
CASE SCREEN.NO = "R30" AND EVENTSOURCE = "R.FIND.LIST"
   BEGIN CASE
      CASE COL = 2
         ED.FILE = FIELD(DBVALUE,'|',1)
         ED.RECORD = FIELD(DBVALUE,'|',2)
         DBCALLURL = 'codeEditor.aspx?fil=':ED.FILE:'&rec=':ED.RECORD
   END CASE
```

You may want to specify the account name in which the file resides. In this case append the name of the account as shown:

```
DBCALLURL = 'codeEditor.aspx?fil=':ED.FILE:'&rec=':ED.RECORD: :"&acc=":ED.ACCOUNT
```

The Code Editor will change to that account before opening the file. The following is an example:

http://192.168.199.194/dbnet/codeEditor/codeEditor.aspx?fil=DBINET&rec=DBI.I.DBIUSERS&acc=DB.NET&ac=garb

Note that you cannot use file names containing '&' such as the '&PH&' file in Universe. In this case set up a q-pointer name that does not contain '&'.

Access denied (code editor)

Note that on the UniData database the access to code editor requires that the loginAccount path in the db.config share the same case as the user start account path in attribute <15> of records in DBIUSERS.

```
<?xml version="1.0" encoding="utf-8" ?>
<dbconfig>
  <entryPoint qcode="">
    <loginHost>127.0.0.1</loginHost>
    <loginHostType>UNIDATA</loginHostType>
    <loginAccount>C:\DesignBais\data\DBINET.DEMO</loginAccount>
    <loginUser>DesignBais</loginUser>
    <loginPassword>DesignBais</loginPassword>
    <loginPublicUser>DesignBais</loginPublicUser>
  </entryPoint>
```

The message "Access denied (code editor)" is displayed where the case does not match, or the user is not in the list of allowed users in web.config.

The HELP menu contains the Shortcuts, About and Preferences options.

There are a large number of Keyboard Shortcuts. Review these to find which ones can be useful to you.

DesignBais Code Editor Keyboard Shortcuts

PC (Windows/Linux)	Mac	Action
Ctrl-Alt-9	Command-Shift-9	Shrink output window
Ctrl-Alt-0	Command-Shift-0	Expand output window
Ctrl-Alt-.	Command-Shift-.	Clear output window
Ctrl-B	Command-B	Insert °
Ctrl-Q	Command-Q	Insert ¶
Alt-Ctrl-N	-	New
Alt-Ctrl-O	-	Open
Alt-Ctrl-V	-	Save
Alt-Ctrl-H	-	Save As
Alt-Ctrl-C	-	Close
Alt-Ctrl-D	-	Delete (file)
Alt-Ctrl-J	-	Compile
Alt-Ctrl-T	-	Account
Alt-Ctrl-P	-	Print preview
Ctrl-Shift-U	Ctrl-Shift-U	change to lower case
Ctrl-U	Ctrl-U	change to upper case
Alt-Shift-Down	Command-Option-Down	copy lines down

Preferences allows you to set and retain your desired Tab size.

Code Editor Locking By-pass

If a program record is left locked after an abnormal termination of a session there is a way to override the lock and open the program in a new instance of the code editor.

As shown below an attempt to open a program that has been locked by another user displays the message in red:

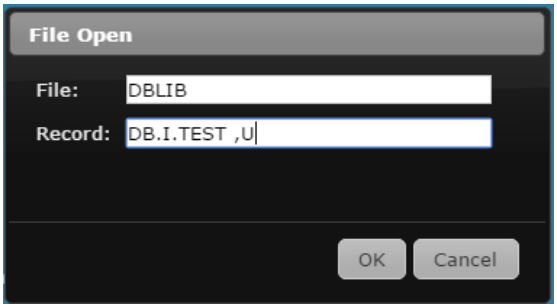
```
The requested record was opened by garb 23 FEB 2017 09:23:52 from account DB.NET use the ,U option to
unconditionally open the record
```

```
1       SUBROUTINE DB.I.TEST
2   *
3   $INCLUDE DBI DBI.COMMON
4   $INCLUDE DBEQU E.DBCLIENT
5   $INCLUDE DBINET E.DBIFORMS
6   $INCLUDE DBINET E.DBIPROP
7   $INCLUDE DBINET E.DBIPARMS
8   *
9       FILENAME = FIELD(SCREENROOT,"_",1)
10      SCREEN.NO = FIELD(SCREENROOT,"_",2)
11      SCREEN.NO = CHANGE(SCREEN.NO,'.DESIGNER','')
12  *
13      THIS.EVENT = PROCESS.EVENT
14      THIS.PARAMETER = PROCESS.PARAMETER
15      EVENTSOURCE = PROCESS.EVENTSOURCE
16      *
17      OPEN 'DBCLIENT' TO F.DBCLIENT ELSE
18         IERR.TEXT = "Unable To Open The File [DBCLIENT]"
19         CALL DBI.G.GLOSSARY(IERR.TEXT)
20         PROCESS.STATUS = 1
21         RETURN
22      END
23  *
24  100:* Event Table
25  *
26      BEGIN CASE
27         CASE THIS.EVENT = "AFTER DISPLAY"   ; * After Screen Display
28            GOSUB AFTER.DISPLAY
29         CASE THIS.EVENT = "BEFORE READ"     ; * Before Read
30            GOSUB BEFORE.READ
31         CASE THIS.EVENT = "AFTER READ"  ; * After Read
32            GOSUB AFTER.READ
33         CASE THIS.EVENT = "VALIDATE"    ; * Validation (on change)
34            GOSUB VALIDATION
```

```
File : DBLIB Record : DB.I.TEST Opened 155 Lines

Unable to gain an exclusive read on the required record. The record is currently locked with exclusive access by garb at
09:23:52 on 23 FEB 2017

The requested record was opened by garb 23 FEB 2017 09:23:52 from account DB.NET use the ,U option to unconditionally
open the record
```

The program record can be opened by appending " ,U" to the record name. The space before the ",U" is important. If it is omitted then you may see the message:

```
Unable to gain an exclusive read on the required record. The record is currently locked with exclusive
access by garb at 09:23:52 on 23 FEB 2017
```



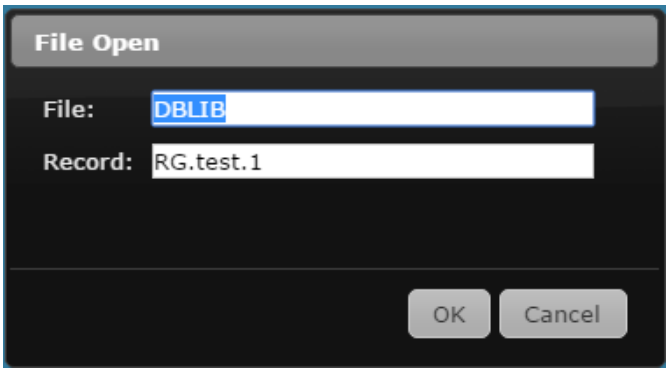
Code Editor Case Sensitivity

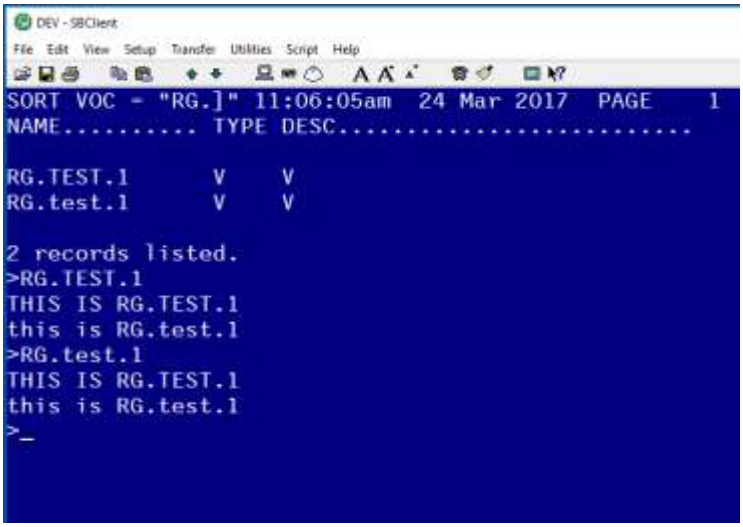
Universe PICK flavour.

The Code Editor File Open process is not case sensitive. In the example below the record with Id of RG.TEST.1 will be retrieved if it exists. So both RG.TEST.1 and RG.test.1 will retrieve the same record. The Id of the record in the program library will have the case of the most recently used Id used in the File Open process.

When you CATALOG the routine the catalog entry in the VOC will be case sensitive and will reflect the case of the record id that you entered. But both RG.TEST.1 and RG.test.1 will run the same routine.

Using the DECATALOG command to remove, say, a lower case VOC entry will in fact remove the object code for the upper case entry.





On other flavors of Universe and on other database platforms you are advised to check for similar behaviour to avoid potential problems.

Code Editor Compiling on UniData

If the code has not been compiled successfully before the following message is displayed:

```
File : DBLIB Record : JL Compiled with Errors at 18:42
ERROR :
Source = CRT OCONV(J,
|_1
[Line 2] Queue empty.
Last line compiled = CRT OCONV(J,
|_1
[Line 2]

No object found for JL
```

The “No object found” is from the CATALOG attempt.

A successful compile and catalog displays this message:

```
File : DBLIB Record : JL Compiled at 18:45
JL Compiled

JL Cataloged
```

A failed compile will recatalog the old object:

```
File : DBLIB Record : JL Compiled with Errors at 18:46
ERROR :
Source = CRT OCONV(J,
|_1
[Line 2] Queue empty.
Last line compiled = CRT OCONV(J,
|_1
[Line 2]

JL Cataloged
```

Code Editor Compiling on D3

DesignBais on D3 uses FlashBASIC object code.

```
File : DBLIB Record : DBI.I.DEMO Compiled at 03:05
DBI.I.DEMO

[820] Creating FlashBASIC Object ( Level 0 ) ...
[241] Successful compile! 36 frame(s) used.
[244] 'DBI.I.DEMO' cataloged
```

```
File : DBLIB Record : DBI.I.DEMO Saved 1740 Lines
```

Note that numeric fields on D3 must be initialised to 0 (zero) not null.

Calling the Code Editor from Designer Process Slots

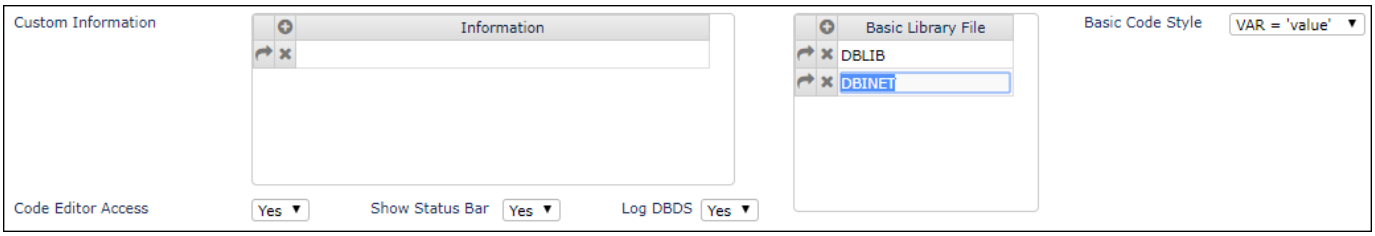
Most *Process* slots now have the tag as a clickable hyperlink. Click this link to access the appropriate tool depending on the content of the process slot.

If the process slot contains a form id *(Filename_Formname)* then the DesignBais Forms Designer or Selection Process tool will open.
If the process slot contains the codeblock designator *C:* then the Code Block maintence form will open.

If the process slot contains the name of an existing subroutine then this routine will be loaded into the Code Editor.

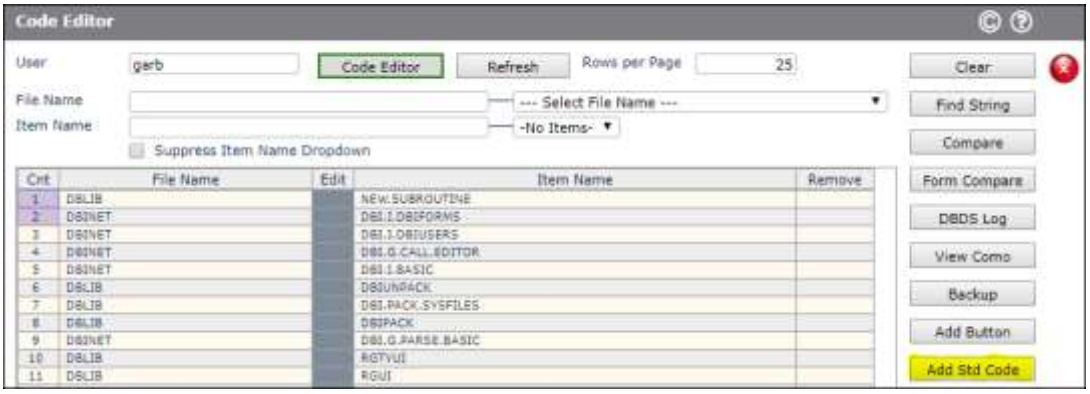
If the process slot is empty then The *Create New Subroutine & Insert Basic Code* Segment form (DBIUSERS_D85) will open to allow the developer to create a new subroutine. The form provides the ability to add basic code for selected DesignBais event processing and as well as code segments from items stored on DBIPARMS.

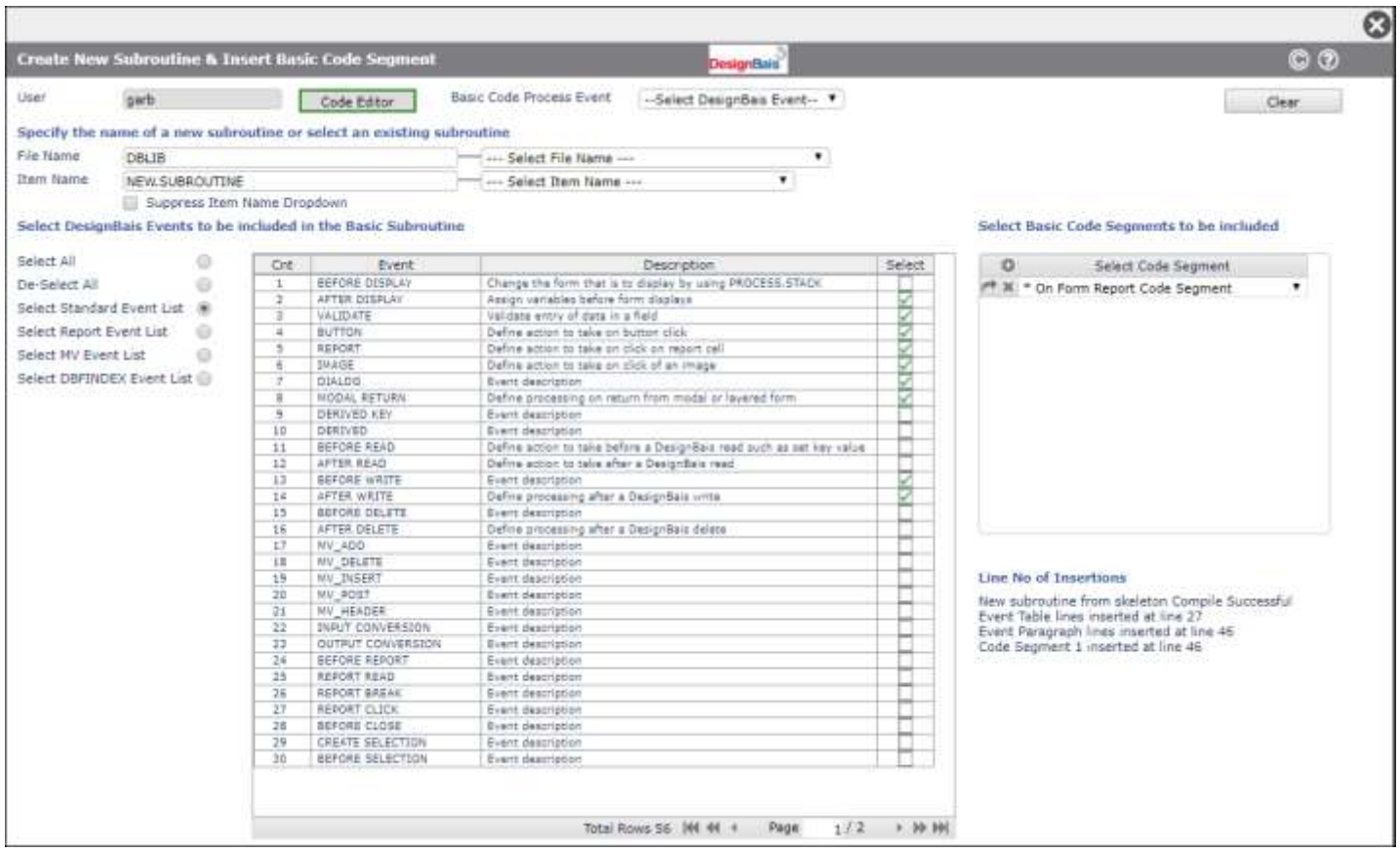
Changes to the user Maintenance form allow developers to designate a list of *Basic Library File*s. The first file in the list becomes the default file name when creating a new subroutine.



The *Basic Code Style* dropdown allows the developer to define the syntax of code inserted using this tool. The developer can choose to have single or double quotes surrounding strings, and a space character surrounding the operand in assignment statements.

The *Create New Subroutine & Insert Basic Code* Segment form (DBIUSERS_D85) is shown below. It is also accessed from the *Add Std Code* button on the Code Editor form:





File Name The name of the Basic Library on which the subroutine exists or on which it is to be created.

Item Name The name of the routine. Existing items can be selected from the dropdown list.

Suppress Item Name Dropdown
For files with a large number of items click this check box to suppress the generation of the dropdown list if the size of the list impacts performance.

Select DesignBais Events to be included in the Basic Subroutine

Select All Selects all the events displayed in the On-form Report.

De-Select All De-select all the events.

Select Standard Event List Selects 9 of the most common events for inclusion in the subroutine.

Select Report Event List Select events that relate to DesignBais Reports.

Select MV Event List Select events that relate to the processing of multivalue grids in Forms Designer.

Select DBFINDEX Event List
Select events related to Word Index definitions.

Select Basic Code Segments to be included

Select Code Segment The dropdown list includes any items from the DBIPARMS file with ids commencing with string *BASIC.SKELETON*. DesignBais includes some code segments as part of the product release. These include:

- BASIC.SKELETON.OFR
- BASIC.SKELETON.POPULATE.DROPDOWN

Developers can create code segments as required and save them in DBIPARMS with ids commencing with string *BASIC.SKELETON* for inclusion in the dropdown list. If you wish to amend the standard DesignBais segments it is recommended that you re-name the amended items so as to avoid losing your changes when DesignBais is upgraded.

Line No of Insertions The line number within the basic subroutine at which event processing paragraphs and basic code segments are inserted. Note that the line number displayed here is the line number at the time of insertion and the actual line number may change as other segments are inserted into the routine. For new subroutines derived from basic code skeleton code the results of the compilation of the generated code are displayed.

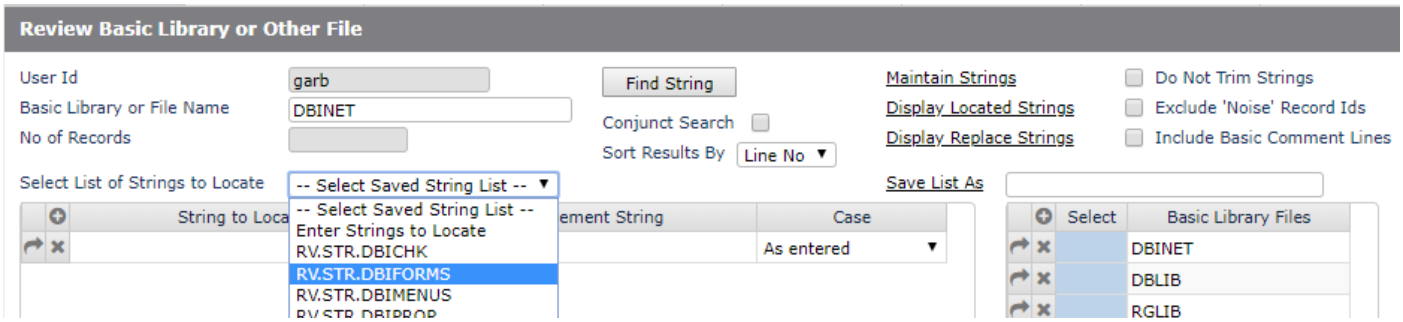
Basic Code Skeleton When a new subroutine is created it will be formed from basic skeleton code from 1 of 3 places in the following hierachy:

- An item in DBIPARMS called *BASIC.SKELETON***weblogon*. This allows all developers to have their own skeleton if required.
- An item in DBIPARMS called *BASIC.SKELETON*. This provides for a skeleton record, in each environment with a unique DBIPARMS file, that will override the DesignBais default.
- From code embedded in the DesignBais form handling subroutine. This is the default.

Find String

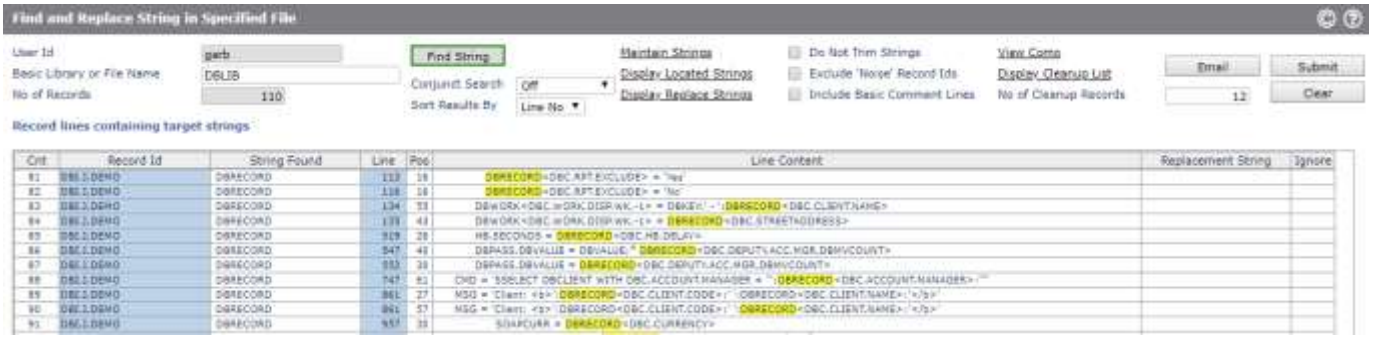
Find String Option - Code Editor Access from Find String form

You can enter the file name to be searched or select from your list of Basic Library Files by clicking blue-highlighted ‘Select’ column.



A default list of strings to locate may display. To create a new list select ‘Enter Strings to Locate’ from the dropdown. You can enter a new list of strings and save the list by entering a name in the ‘Save List As’ field then clicking the ‘Save List As’ button. Lists are saved on DBIPARMS with a key commencing with ‘RV.STR.’. The list is also saved when Submit is clicked. If you have not specified an existing saved list name, and the ‘Save List As’ field is null then on submit the list of strings will be saved under your userid (WEBLOGON) on the DBIPARMS record ‘RV.STR.userid’.

Click the ‘Find String’ button once you have the strings to be located, and associated replacement strings (optional) in place.



The strings are located and displayed on an On-form Report. Clicking on the blue-highlighted ‘Record Id’ column will take you into the Code Editor with the selected routine open for editing.

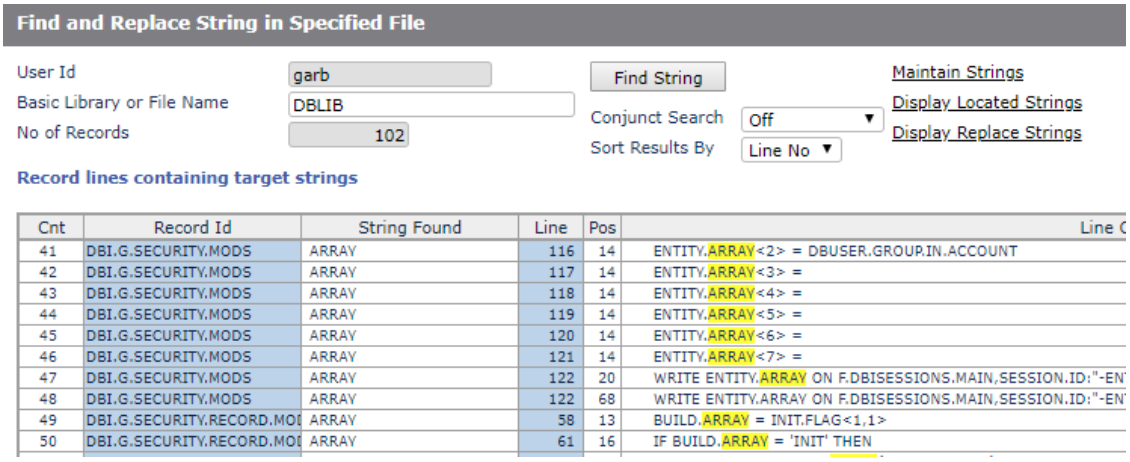
Results can be sorted by either line number, or by target string, within each routine in which a string is located.

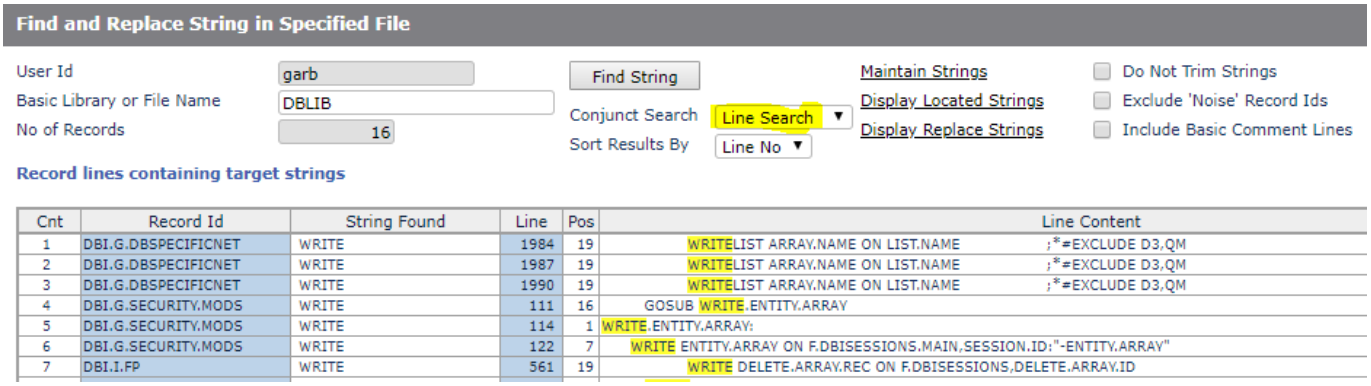
The *Conjunct Search* check box allows for the results of a search to be used as the starting point for the next search. After the initial search results are displayed select the required option and enter the strings that are to be searched for within the list of records returned by the first search. Conjunct options must be selected **after** the *Find String* button is clicked with the *Conjunct Search* set to *Off* and a list of record lines containing target strings is displayed.

The *Conjunct Search* options are:

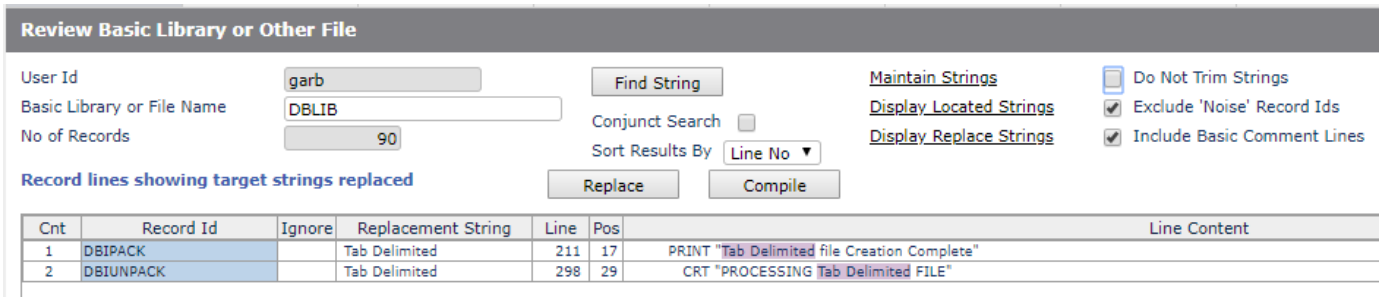
Item Search The search for target strings is limited to only the **items** returned by the previous search.
Line Search The search for target strings is limited to only the **lines** returned by the previous search.
Word Search The search for target strings is limited to only the **lines** returned by the previous search and the target strings must exist as distinct words rather than as part of a longer string in order to be selected for display. So FIELD will be selected from the string (FIELD, but not from FIELD.LIST or FONTFIELD.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3





Click the 'Display Replace Strings' button to display the located lines with the replacement string. If the file being searched is in the list of Basic Library Files then the 'Compile' button will appear.



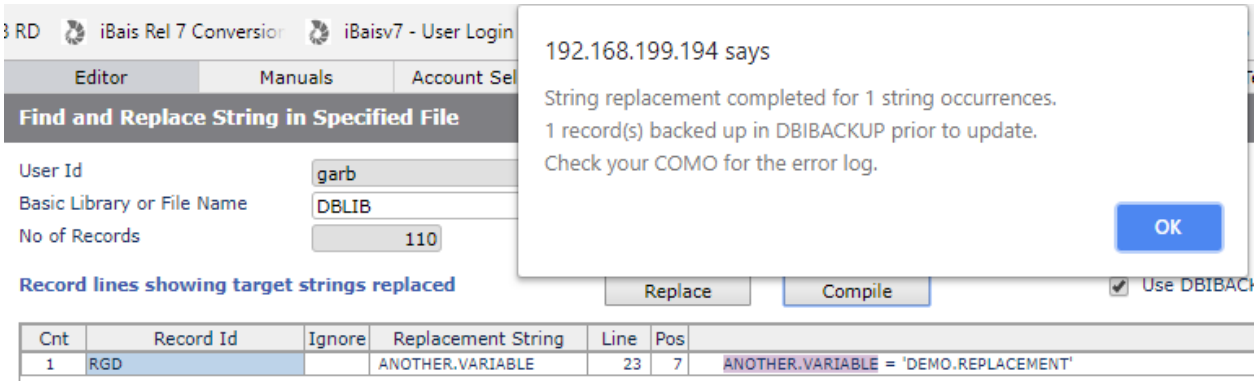
Click the 'Replace' button to update the records with the replacement string, or just exit if you are using this form as a find and display tool.

In the following example the string ‘RGVARIABLE’ is to be replaced by the string ‘ANOTHER.VARIABLE’.

Click the ‘Ignore’ column in the row associated with any routine where you do not want the replacement to occur.

Click ‘Replace’. The message is displayed to allow you to confirm the update of all listed programs.

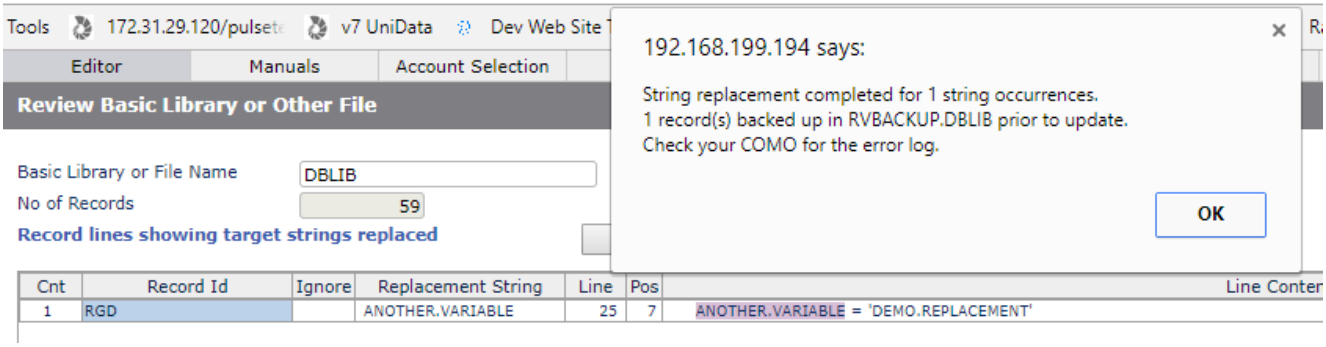
Clicking ‘Yes’ confirms the update. The confirmation message that displays is shown in the image below.



This confirms how many records were updated and advises that a copy of all updated routines, prior to update, has been stored in the DBIBACKUP file based on the fact that the check box *Use DBIBACKUP for pre-change backup* has been ticked.

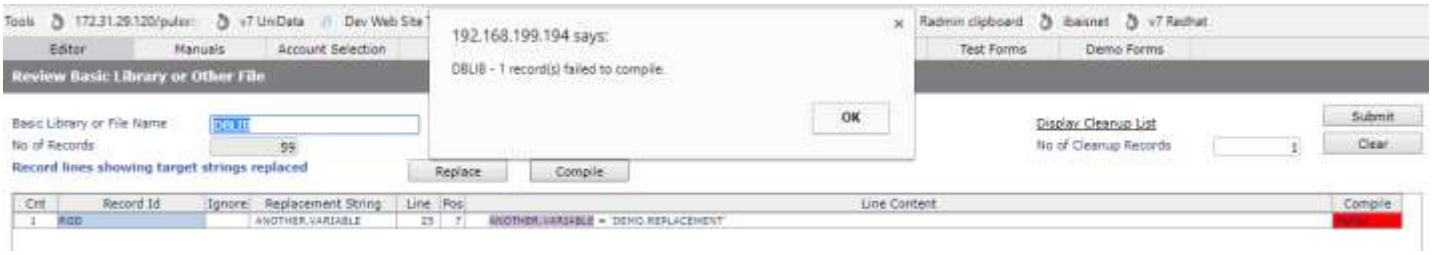
Using DBIBACKUP allows you to access the backed up items using the *Backup* function.

If not ticked then items are backed up in a file called ‘RVBACKUP.filename’ where ‘filename’ is the name of the searched file.

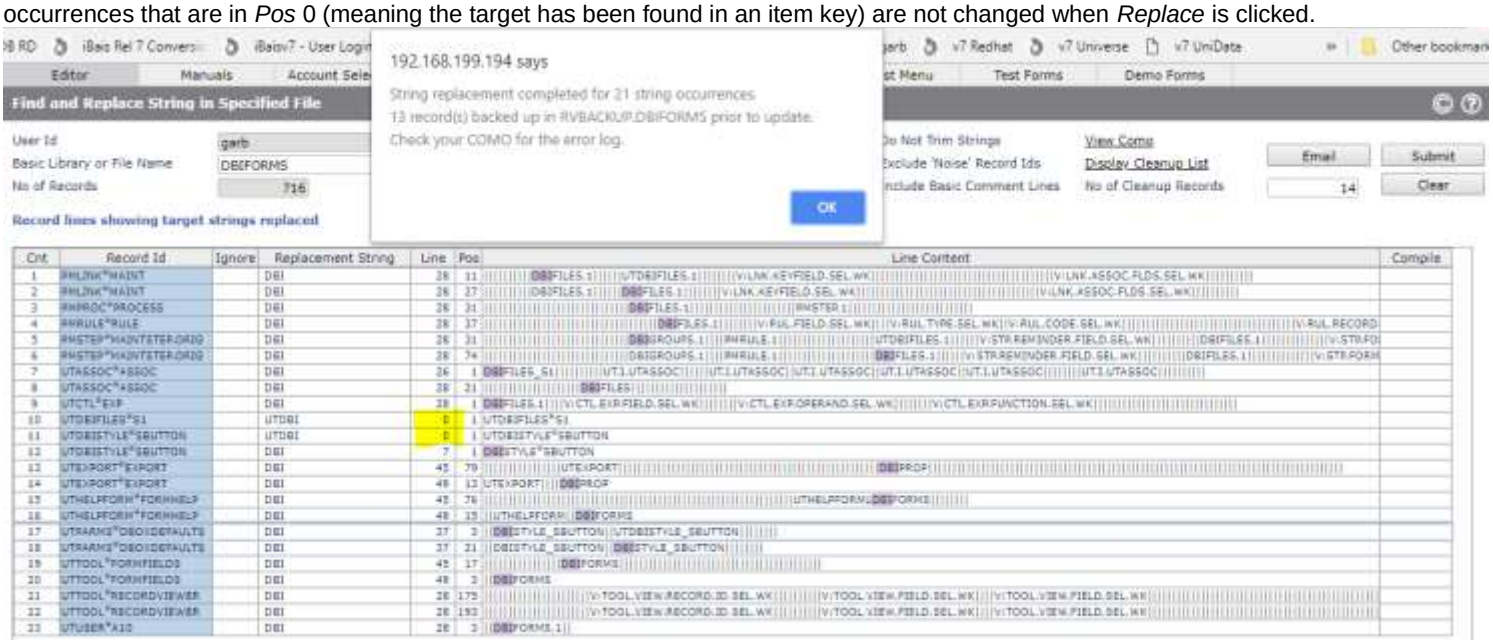


Check your como to view any messages generated. The como will report cases where the update was not finalised because the record could no longer be found, or the record was locked by another user.

You can compile all displayed routines by clicking the ‘Compile’ button. The green ‘Compiled’ message indicates a successful compile. A red ‘Failed’ message will appear if the compilation is not successful.



The string replacement does not occur if the target string is part of the key of a record. Note in the following example that the 2 occurrences that are in *Pos* 0 (meaning the target has been found in an item key) are not changed when *Replace* is clicked.



The list of ‘Suffix to Cleanup’ is designed to allow you to enter commonly used suffixes when naming backup copies of programs or other file records. Records in the searched file that have names ending in one of these suffixes, or ending in a date will be loaded into the cleanup list.

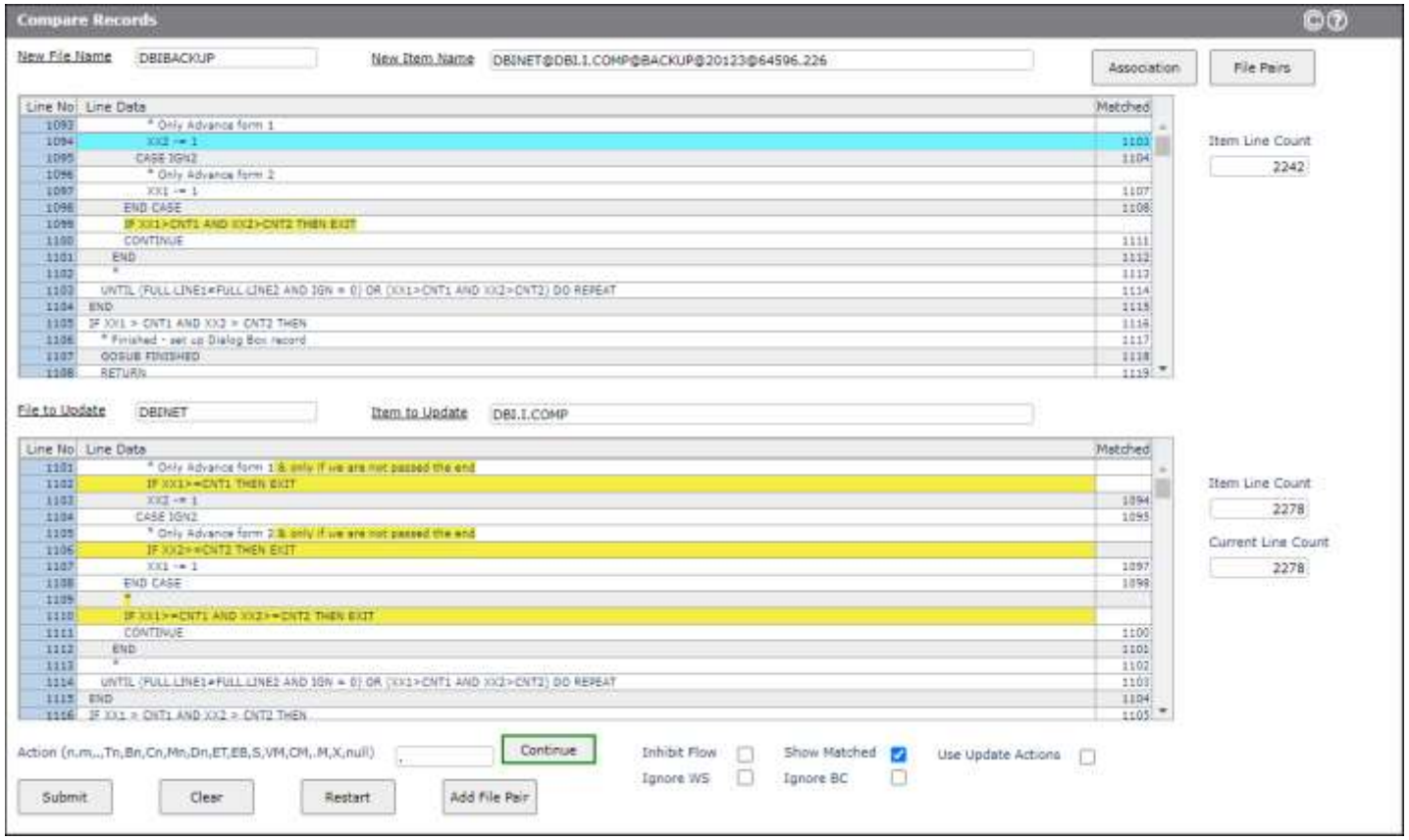
The ‘Display Cleanup List’ button displays the list of records that may be cleanup up. Click the ‘Keep’ column against those that you do not want to be moved out of the file.

Click the ‘Move Records’ button to move all records not flagged as ‘Keep’ to a file named ‘RVCLEANUP.filename’ where ‘filename’ is the name of the searched file.

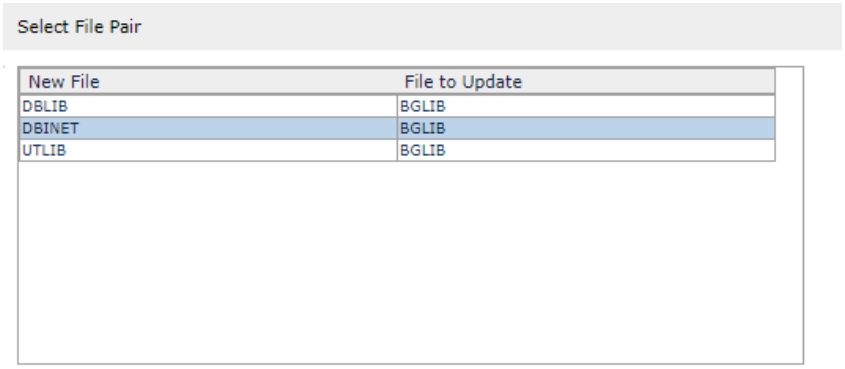
Compare Option

Compare Option

Allows line by line comparison of 2 records. The record displayed at the bottom of the form, the *File to Update* record, can be modified by merging, or copying lines from the top *New File Name*. Lines can also be deleted. The *Action* codes, described in the help, are detailed below.



The *File Pairs* button displays the list of file pairs saved via the *Add File Pair* button. Clicking *Add File Pair* saves the currently displayed *New File Name* and the *File to Update* as a associated pair. This feature allows the developer to quickly recall commonly used file pairs.



The *Inhibit Flow* check box causes the display of both records to remain fixed after applying a change, such as merging a line into the File to Update record

The *Show Matched* check box displays the line number of the record in the *New File Name* file that matches a line in the *File to Update* file. If there is no matching line then row in the Matched column is null.

The *Update Defaults* check box changes the default setting of the Action field. When unchecked the compare works by defaulting the required action to a "." (a period symbol). This serves to allow the user to see all differences, highlighted in yellow, then click *Continue* to move the compare to the next difference. The user can progress through the entire record, seeing all differences, without making any changes to the *Item to Update*. So the *Update Defaults* check box changes this behaviour to a behaviour designed to facilitate the applying of changes to the *Item to Update* so as to bring it into line with the top record. If the compare notes that there are two new lines in the top record then the default action will be *M2*. Clicking *Continue* with action of *M2* causes the top 2 lines of the top record to be merged into the bottom record, just before the line displayed at the top of the bottom display.

Action Codes:

n.m	Restart comparison with top at line n and bottom at line m.
.	Find next difference
Tn	Increment top by n lines (or decrement if n negative)
Bn	Increment bottom by n lines (or decrement if n negative)
Cn	Copy n lines from top to overwrite n lines at bottom
Mn	Merge n lines from top before top line of bottom
Dn	Delete n lines from bottom
ET	Edit top program and return
EB	Edit bottom program and return
S	Display all of the top line of both
VM	View Multivalue Association
CM	Copy Multivalue Associated Attributes
.M	Move past Associated Multivalued Attributes
X	Exit
Null	No entry will advance top and bottom by 10 lines

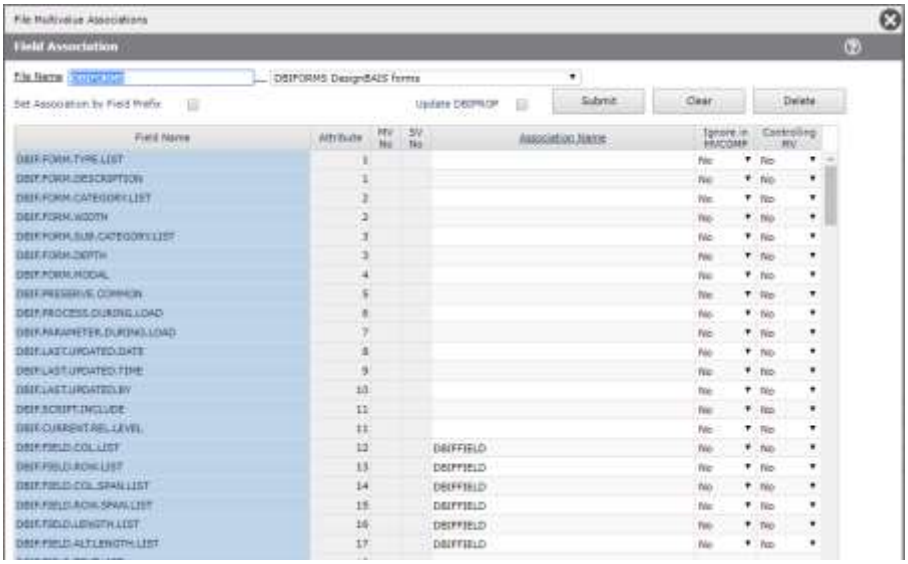
If changes are made to the bottom record, the *Item to Update*, then clicking the *Submit* button will save those changes on the *File to Update*. Clicking the *Restart* button will discard any changes made to the *Itemto Update*.

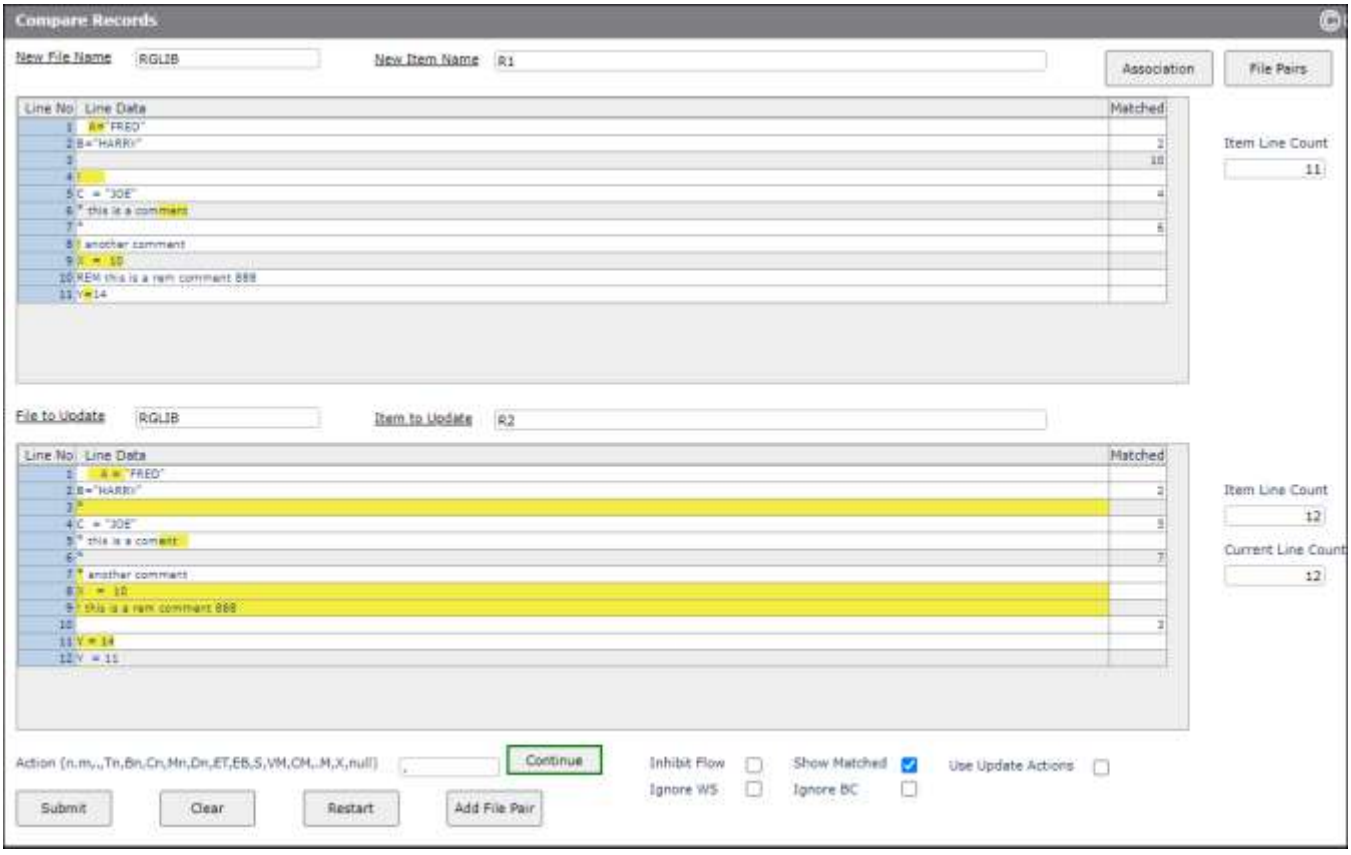
Clicking a row in the *Line No* column (highlighted in blue) of either the top or bottom display will flag that row as the start row for the comparison. The *Action* field will be updated with a value of either (top) *Row.* or (bottom) *.Row*. After clicking a row in the top and in the bottom display the *Update* field will be set to *TopRow.BottomRow* ready for the *Continue* button to be clicked.

The *Ignore WS* check box allows you to ignore whitespace differences. All lines have all spaces removed before the compare. This means that "A = B" will match "A=B".

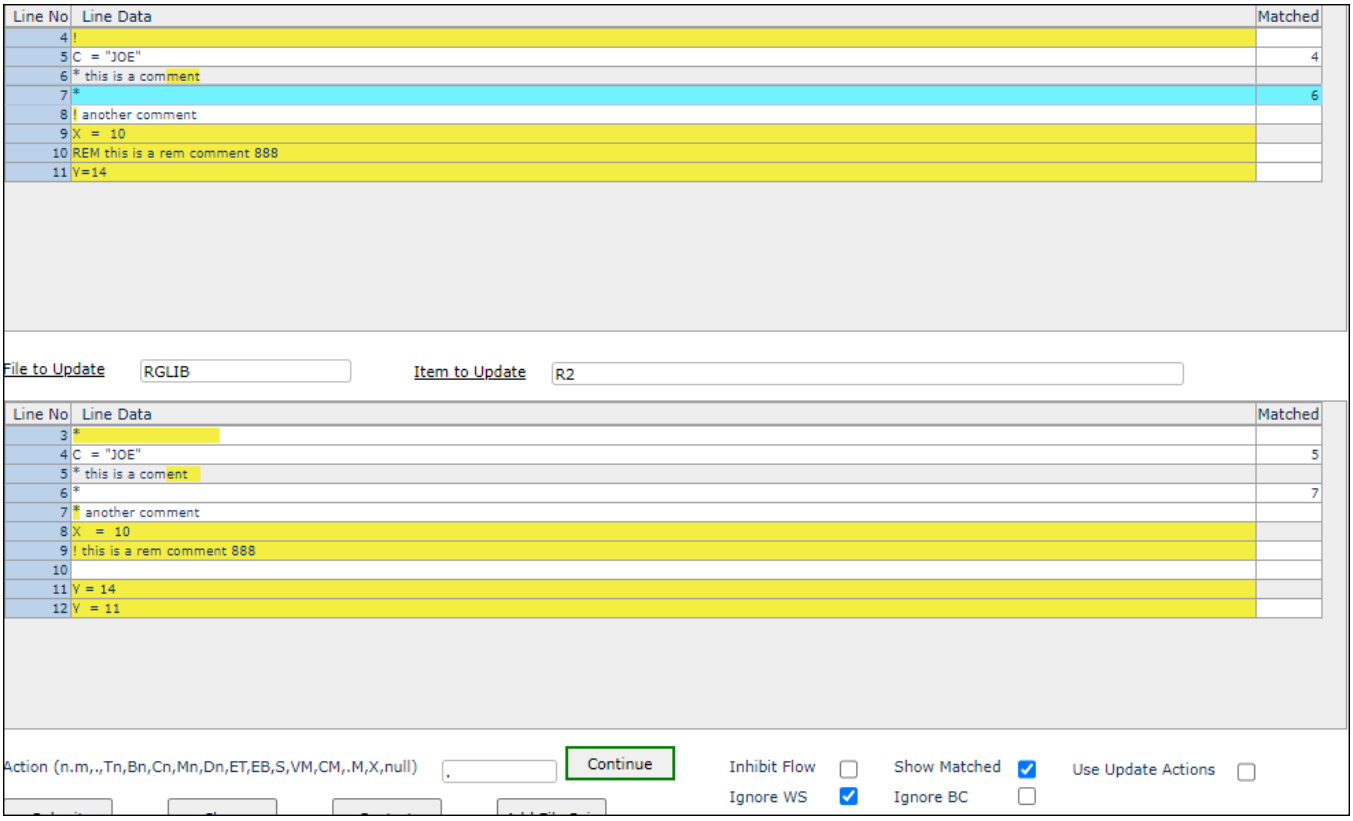
The *Ignore BC* check box allows you to ignore all basic code comment lines. Lines are considered comment lines if the line starts with any of the following strings: "*", "!", "REM ", "$*".

The *Association* button displays a form to allow the definition of associated attributes for a file. These definitions are used in the Multivalue Compare Form.

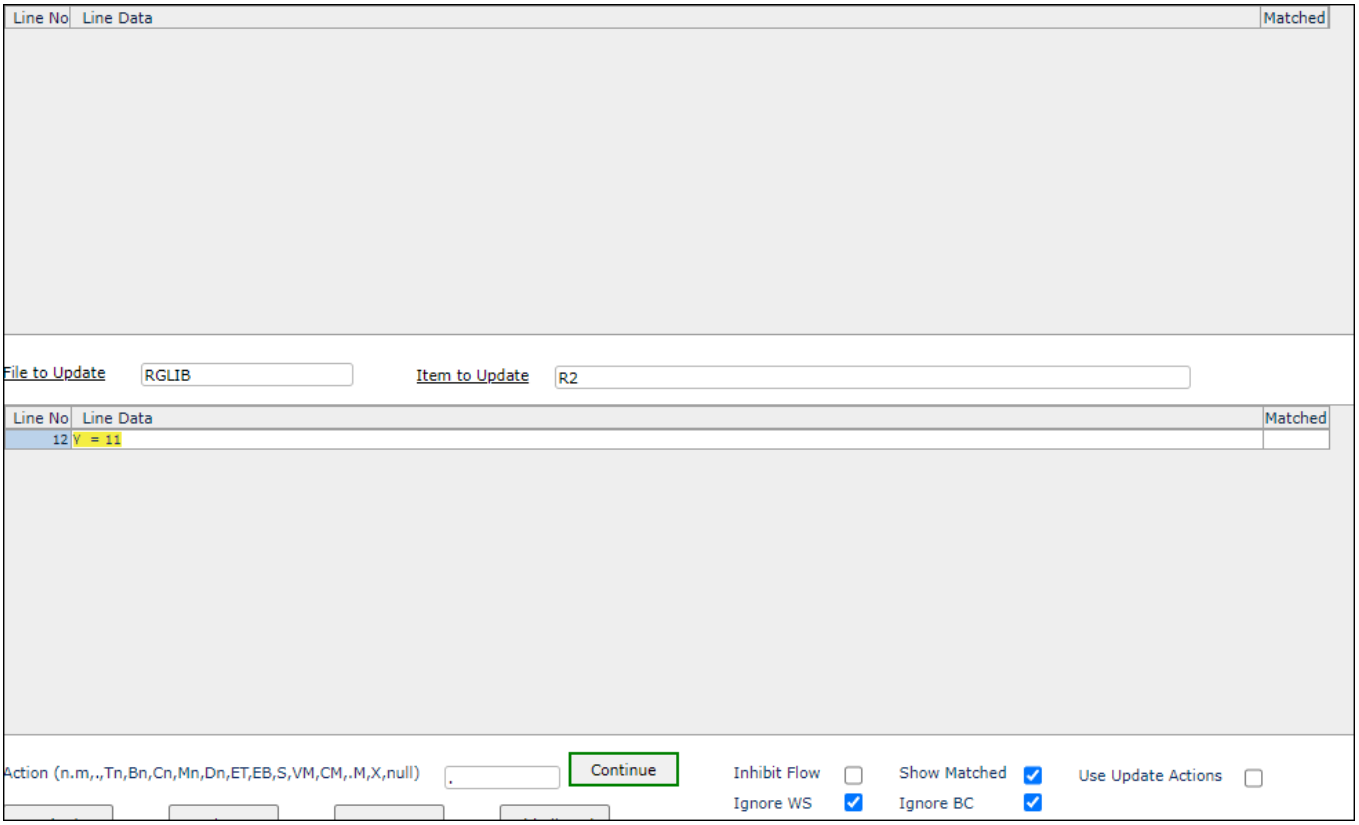




The effect of the *Ignore WS* and *Ignore BC* is shown here. Above neither check box is checked.
Below, when whitespace is ignored, the list of differences is reduced.



When the comment lines are ignored the only significant difference is revealed.

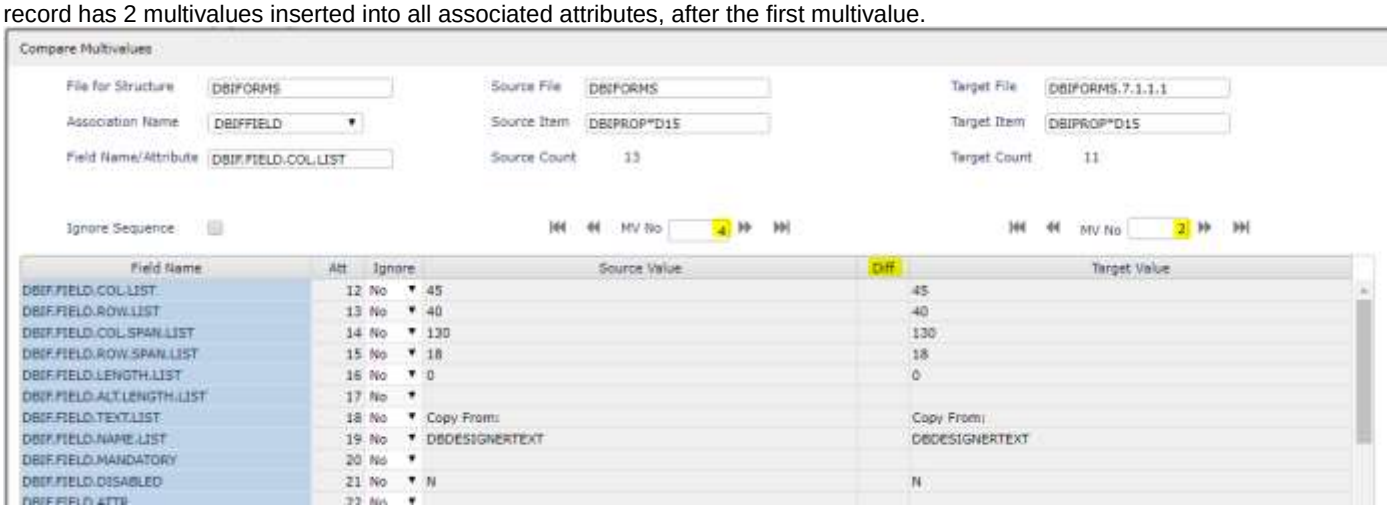


Multi-value Compare Option

Clicking the update option option VM in the Compare Form, when comparing an attribute with multivalues, opens the MVCOMP form. This allows the comparison of the values in the associated set of attributes.

In the first screenshot the top record now displays on the left. The *MV No* field displays the multivalue number that does not match the corresponding multivalue in the bottom record, now displayed on the right. In this example the first multivalue in the DBIF.FIELD.COL.LIST attribute (see blue highlighted column on the left) has the same value in both records. But the value in the second multivalue differs. So the display sits with this mismatch visible. Note the # symbols in the *Diff* column indicating that the value on the left is not equal to the value on the right.

By using the scroll arrows the left hand display can be advanced from multivalue 2, through 3, to 4, at which point the display reveals that multivalue position 4 on the left matches, in all attributes, the multivalue position 2 on the right. This indicates that the left hand record has 2 multivalues inserted into all associated attributes, after the first multivalue.



This is shown in the *Matched Values* grid on the right. Multivalue 1 matches in *Source* and *Target*.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Matched Values 11

Value	Source	Target
1	1	1
2		4
3		5
4	2	6
5	3	7
6	4	8
7	5	9
8	6	10
9	7	11
10	8	12
11	9	13
12	10	
13	11	

DBDS Log

DBDS Log Option

To assist the developer in debugging an application it may be helpful to retain the DBDS output from various subroutines. To this end the DBDSVIEW form provides a means of displaying the accumulated DBDS output.

The developer can turn the feature on and off within the DBDSVIEW form. The flag to control this feature is held on the DBIUSERS file and this option can also be toggled on and off in the *Users* maintenance form.

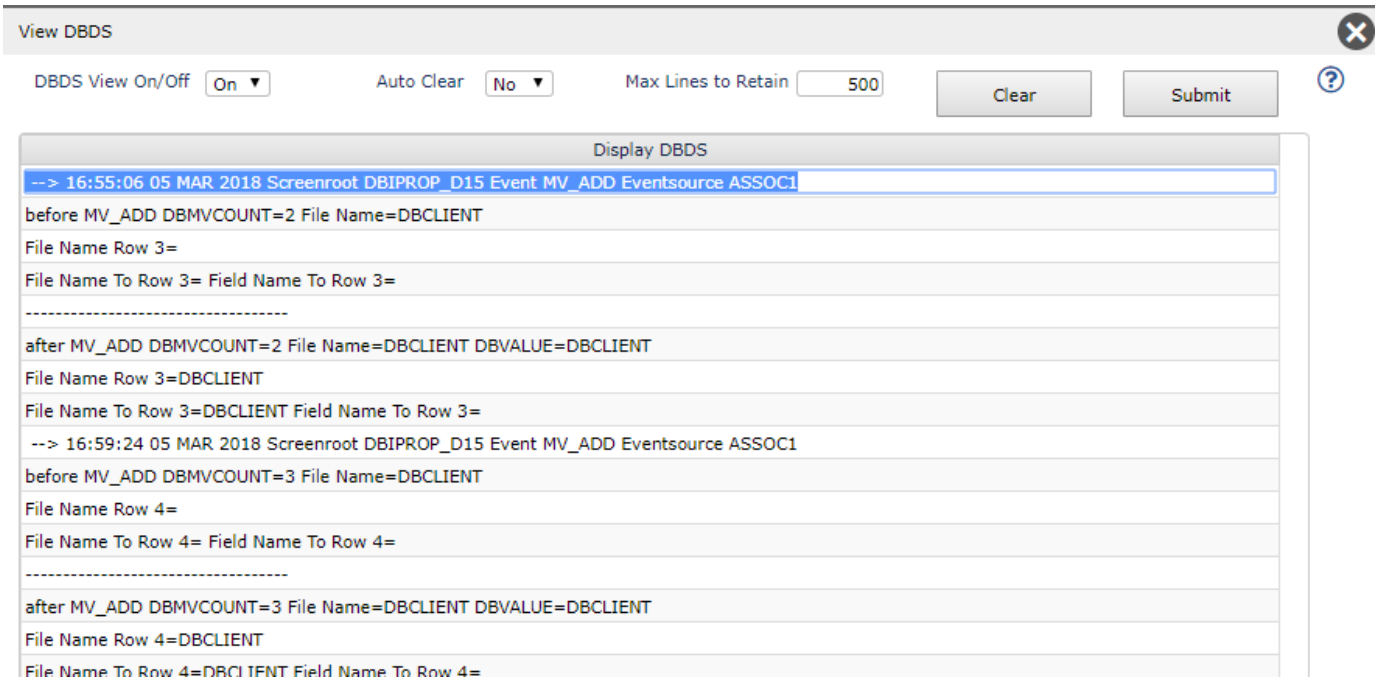
Auto Clear allows the accumulated DBDS output to be cleared automatically when the number of lines exceeds the value entered in the *Max Lines to Retain* field.

Alternatively use the *Clear* button to clear the list.

Clicking the *Submit* button will save any changes made to the parameters.

Each event that triggers a DBDS display will add a “Header” line to the accumulated DBDS log showing the time and date and the value in SCREENROOT, PROCESS.EVENT and PROCESS.EVENTSOURCE for the event. IERR.TEXT is also saved in the log.

There is a maximum of 2000 lines per user that can be retained in the log regardless of the setting in *Max Lines to Retain*. The log is stored in the DBIPARMS file with record id of *DBDS*weblogon* where *weblogon* is the DesignBais user id for the session.



Chapter 28 – Phantom Processing

Phantom Processing

To assist developers to set up phantom processing the following example demonstrates some of the features that can be used in DesignBais.

A method of tracking phantom processes is available. This uses the file DBISTATS to hold a record containing details of a phantom process and is invoked by utilising the “PHANTOM_TRACK” function within DBI.G.DBSPECIFICNET. An example of the required code is shown below. In this example the program that is to be run as a phantom is DBI.I.UPGRADE.

Calling DBI.G.DBSPECIFICNET will create a record in the file DBISTATS with a key commencing with the string ‘PHANTOM_’. In the example below the PH.ID is set up to include this string and this is the recommended approach. If this string is omitted the DesignBais routine will prefix the parameter that is passed via OS.DATAIN with this string.

The DBISTATS record contains the following fields:

<1> Phantom PID
<2> Command run
<3> Approximate start time
<4> Start Date
<5> User Login
<6> Progress Percentage (initialised to zero)
<7> Progress Message or Error Message when Failed (initialised to 'Started')
<8> Account where run
<9> Active Form File
<10> Active Form Name
<11> Active Form Description
<12> Status (initialised to 'S') – valid entries are: S=Started, A=Active, C=Complete, F=Failed

The developer can use the DBTIMER function to monitor the progress of the phantom. The phantom program should delete this DBISTATS record when the phantom completes. The DesignBais form handling routine that is called to handle the TIMER event can test for the presence of the DBISTATS record in order to determine that the phantom process has ended.

The DBWORK variable DBIPM.U.PROGRESS.WK is used to display the progress bar indicating the progress of the phantom program.

Within the phantom program it is necessary to calculate the progress of the processing and update a record on the database such that on each DBTIMER event the progress bar can be updated with the percentage of processing completed (see PH.PERC variable in the example code below).

```
* Set the phantom id and initiate the phantom program DBI.I.UPGRADE
    PH.ID = 'PHANTOM_':SESSION.ID:'_':DATE():'_':TIME()
    OS.FUNCTION = 'PHANTOM_TRACK'
    OS.DATAIN = 'DBI.I.UPGRADE  ':PH.ID
    OS.DATAIN<2> = PH.ID
    OS.DATAOUT = ''
    CALL DBI.G.DBSPECIFICNET(OS.FUNCTION,OS.DATAIN,OS.DATAOUT)
* phantom process id extracted in DBI.G.DBSPECIFICNET...otherwise an error
    IF NUM(OS.DATAOUT) THEN
        * phantom started
      DBTIMER = DBTIMER.INCREMENT
      DBTIMER<1,3> = "TIMER"
      DBTIMER<1,4> = "DBI.I.UPGRADE"
      DBTIMER<1,5> = "PHANTOM_TRACK_":PH.ID
      PH.PERC = 0
      PH.MSG = 'Phantom Started'
      PGS = '<progress class="dbaisSearchLabelBlue" value="':PH.PERC:'" style="width:200px;"></progress>'
      DBWORK<DBIPM.U.PROGRESS.WK> = PGS
      DBWORK<DBIPM.U.PROGRESS.MSG.WK> = PH.MSG
    END ELSE
        * phantom program failed - pass back error & cleanup
      IERR.TEXT = OS.DATAOUT
    END
```

Note that the OS.FUNCTION of ‘PHANTOM_TRACK’ creates a record on the file DBISTATS with a status of ‘S’ (Started). The developer’s phantom routine should capture the Id used as the key to the DBISTATS record (PH.ID in the example) and update the status to say ‘A’ (Active) by writing to the DBISTATS file DBISA.PH.STATUS attribute.

The developer can determine the progress of the phantom process by any desired method but the example below demonstrates a simple approach. The number of records selected is captured at the start. As each record is processed a counter is incremented and at intervals the DBISTATS record can be updated with the progress percentage which is simply a number in the range 0 to 100. Refer to the code snip below. Note that the status must be updated by the developer, changing the ‘S’ to ‘A’.

```
UPDATE.PHANTOM.PROGRESS:
    IF DBWORK<DBIPM.U.SELCNT.WK>+0 = 0 THEN
      PH.PERC = 100
    END ELSE
      PH.PERC = INT(DBWORK<DBIPM.U.CNT.WK>*100/DBWORK<DBIPM.U.SELCNT.WK>)
    END
    READ PH.REC FROM F.DBISTATS,PH.ID ELSE PH.REC = ''
    PH.REC<DBISA.PH.PERCENTAGE> = PH.PERC
    PH.REC<DBISA.PH.MESSAGE> = UPGRADE.OPTIONS<OPTNO>
    PH.REC<DBISA.PH.SELECTED>= DBWORK<DBIPM.U.SELCNT.WK>
    PH.REC<DBISA.PH.PROCESSED>= DBWORK<DBIPM.U.CNT.WK>
    PH.REC<DBISA.PH.STATUS>= 'A'
    WRITE PH.REC ON F.DBISTATS,PH.ID
    RETURN
```

The developer must delete the DBISTATS record at the end of the phantom process. The absence of this record can then be used in the TIMER event processing to determine that the phantom has completed.

Phantom processes create an entry in the &PH& file in Universe (_PH_ in UniData). These records can be removed using the DesignBais function “CLEAR.PHANTOM”. The developer must supply the same prefix that was used when starting the phantom with the “PHANTOM_TRACK” function. This can be done in the TIMER event processing as shown below.

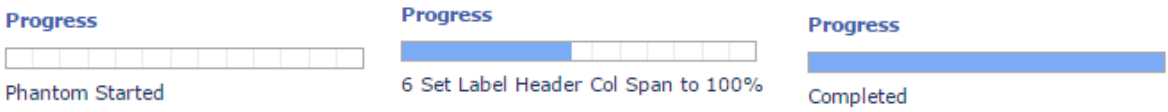
```
TIMER:
  BEGIN CASE
    CASE EVENTSOURCE[1,13] = "PHANTOM_TRACK" AND SCREEN.NO = "U1"
          PH.ID = EVENTSOURCE[15,LEN(EVENTSOURCE)-14]
          READ PH.REC FROM F.DBISTATS,PH.ID THEN
              DBWORK<DBIPM.U.CNT.WK> = PH.REC<DBISA.PH.PROCESSED>
              DBWORK<DBIPM.U.SELCNT.WK> = PH.REC<DBISA.PH.SELECTED>
              IF DBWORK<DBIPM.U.SELCNT.WK> = 0 THEN
                PH.PERC = 100
              END ELSE
                PH.PERC = 100 * DBWORK<DBIPM.U.CNT.WK>/DBWORK<DBIPM.U.SELCNT.WK> '0'
              END
              PH.REC<DBISA.PH.PERCENTAGE> = PH.PERC
              PH.MSG  = PH.REC<DBISA.PH.MESSAGE>
              WRITE PH.REC ON F.DBISTATS,PH.ID
              PGS = '<progress class="dbaisSearchLabelBlue" value="':PH.PERC/100:'" style="width:200px;"></progress>'
              DBWORK<DBIPM.U.PROGRESS.WK> = PGS
              DBWORK<DBIPM.U.PROGRESS.MSG.WK> = PH.MSG
          END ELSE
              * Record deleted = finished
              PH.PERC = 100
              PH.MSG = 'Completed'
              DBTIMER = 0
              PGS = '<progress class="dbaisSearchLabelBlue" value="':PH.PERC/100:'" style="width:200px;"></progress>'
              DBWORK<DBIPM.U.PROGRESS.WK> = PGS
              DBWORK<DBIPM.U.PROGRESS.MSG.WK> = PH.MSG
              OS.FUNCTION = 'CLEAR.PHANTOM'
              OS.DATAIN = 'DBI.I.UPGRADE'
              CALL DBI.G.DBSPECIFICNET(OS.FUNCTION,OS.DATAIN,OS.DATAOUT)
          END
  END CASE
```

The phantom record in the &PH& file is created when the phantom program is initiated.

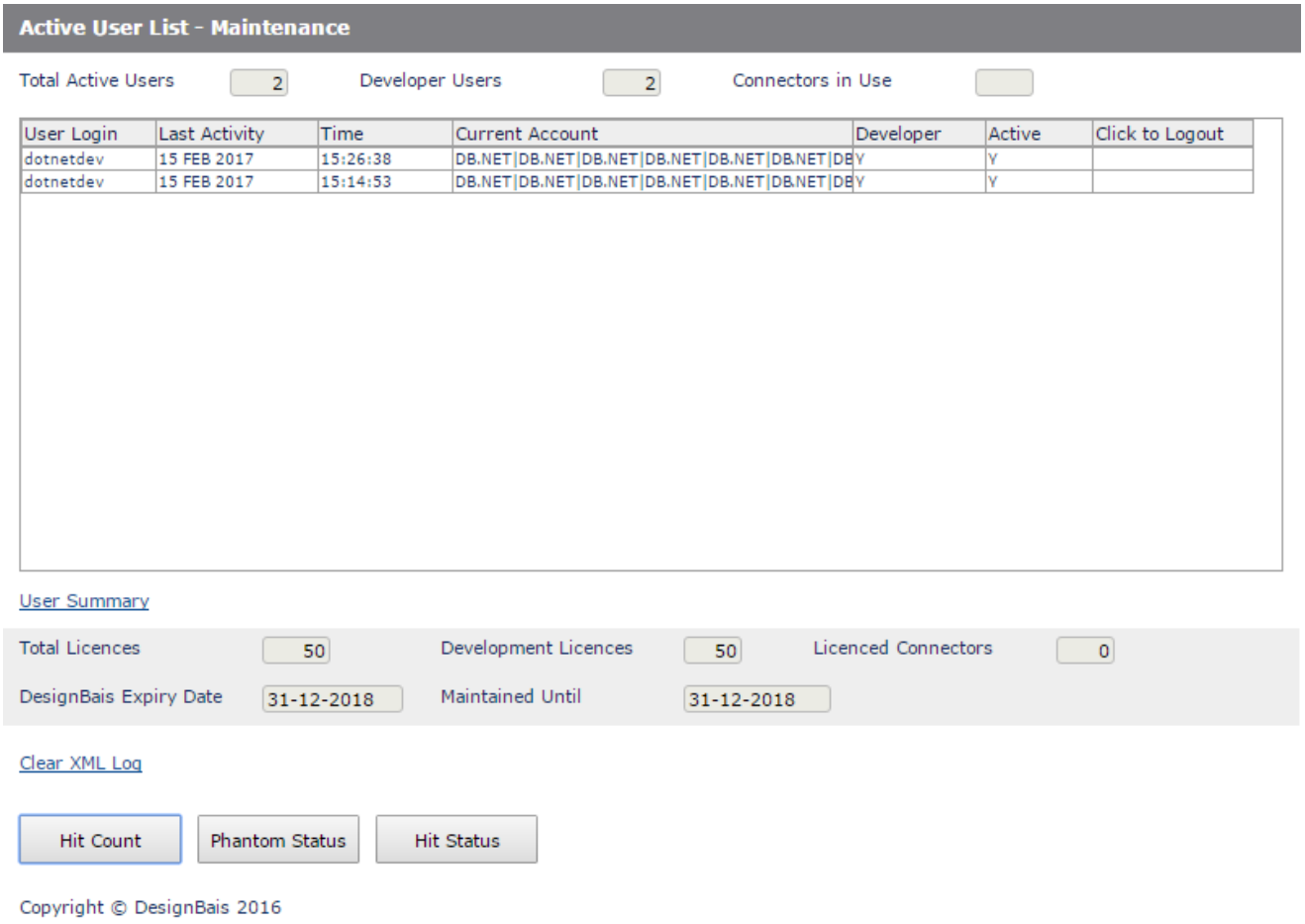
```
SORT &PH& 02:22:36pm  01 Feb 2017  PAGE    1
&PH&.........................

DBI.I.UPGRADE_51752_17930
DBPURGESESSIONS_33565_17930
DBPURGESESSIONS_33608_17929
```

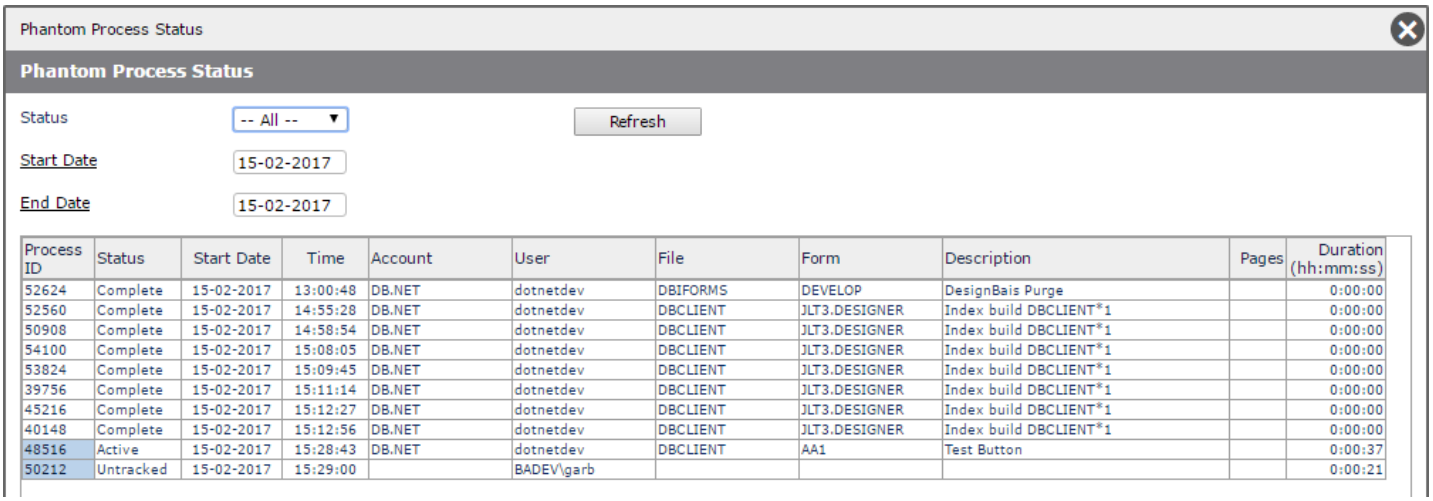
The DesignBais function “CLEAR.PHANTOM” called from DBI.G.DBSPECIFICNET will clean up these records.

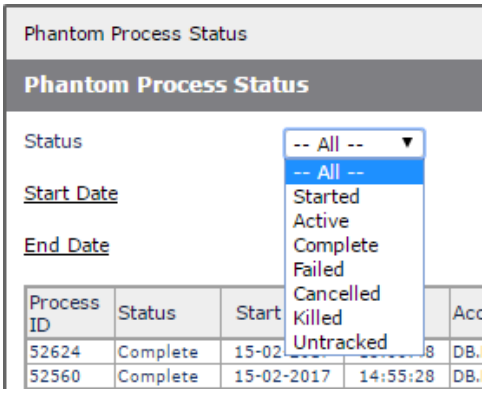
Using a DBWORK field as shown in the code above displays the progress bar like this. The description below the progress bar is displayed in another DBWORK field.



The currently running phantom processes can be displayed using the Phantom Status button on the Active Users form. If the phantom processes were intiated via the “PHANTOM_TRACK” function within DBI.G.DBSPECIFICNET an appropriate status will display otherwise the status is ‘Untracked’.







The phantom process can be killed by clicking column 1 of a row if the Process ID cell is highlighted. Only tracked phantoms, once killed, will display with a status of 'Killed'. This is because the display is based on the records that are written to DBISTATS. Untracked phantoms, be definition, do not have a DBISTATS record and therefore once killed these entries disappear.

All untracked phantom processes will display, and can be killed, regardless of the user that initiated them. For tracked phantoms, however, only those initiated by the current DesignBais user (WEBLOGON) will display.

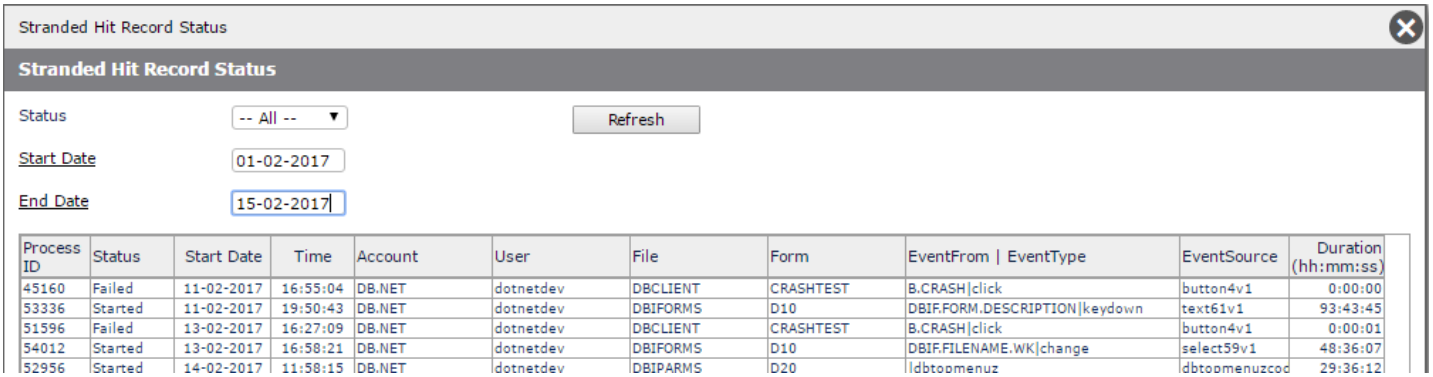
The records in DBISTATS, and in the &PH& (_PH_) file are purged by the DesginBais purge sessions routine DBI.P.PURGENET. Records are retained for the number of days set in the System Parameters Phantom Log Days field.

Note that on D3 the functionality to kill the process id has not been implemented.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Hit Status

The Hit Status button on Active users displays records from DBISTATS that record hit details. The status can be Started, Failed or Timed Out.



Process ID	Status	Start Date	Time	Account	User	File	Form	EventFrom | EventType	EventSource	Duration (hh:mm:ss)
45160	Failed	11-02-2017	16:55:04	DB.NET	dotnetdev	DBCLIENT	CRASHTEST	B.CRASH|click	button4v1	0:00:00
53336	Started	11-02-2017	19:50:43	DB.NET	dotnetdev	DBIFORMS	D10	DBIF.FORM.DESCRIPTION|keydown	text61v1	93:43:45
51596	Failed	13-02-2017	16:27:09	DB.NET	dotnetdev	DBCLIENT	CRASHTEST	B.CRASH|click	button4v1	0:00:01
54012	Started	13-02-2017	16:58:21	DB.NET	dotnetdev	DBIFORMS	D10	DBIF.FILENAME.WK|change	select59v1	48:36:07
52956	Started	14-02-2017	11:58:15	DB.NET	dotnetdev	DBIPARMS	D20	ldbtopmenuz	dbtopmenuzcod	29:36:12

Normally there should be no hit records with a status of ‘Started’. If the Process ID is still in the LISTU then ‘Started’ hits are filtered out of the display.

If the server hit fails, such as when a call is made to a non-existent or uncatalogued subroutine, then the status will be set to ‘Failed’. If the server hit causes a program loop then the status will be set to ‘Timed Out’. If highlighted the Process can be killed.

On Universe and UniData only DesignBais user processes are displayed. This is achieved by filtering out the “telnet” entries from the LISTU display.

On D3 the LISTU display does not distinguish between DesignBais sessions and telnet sessions so the Hit Status will display both types of session for the DesignBais connection user.

It is therefore recommended, on D3, that the DesignBais connection user is reserved for DesignBais and not used for telnet sessions. By implementing this policy the Hit Status display will then only reflect DesignBais sessions.

Note that on D3 the functionality to kill the process id has not been implemented.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Chapter 29 – Header Form Overlay

Header Form Overlay

Header Form Overlay describes the concept of invoking a header form in which the form depth is set to zero and most of the elements on the form are initially hidden.

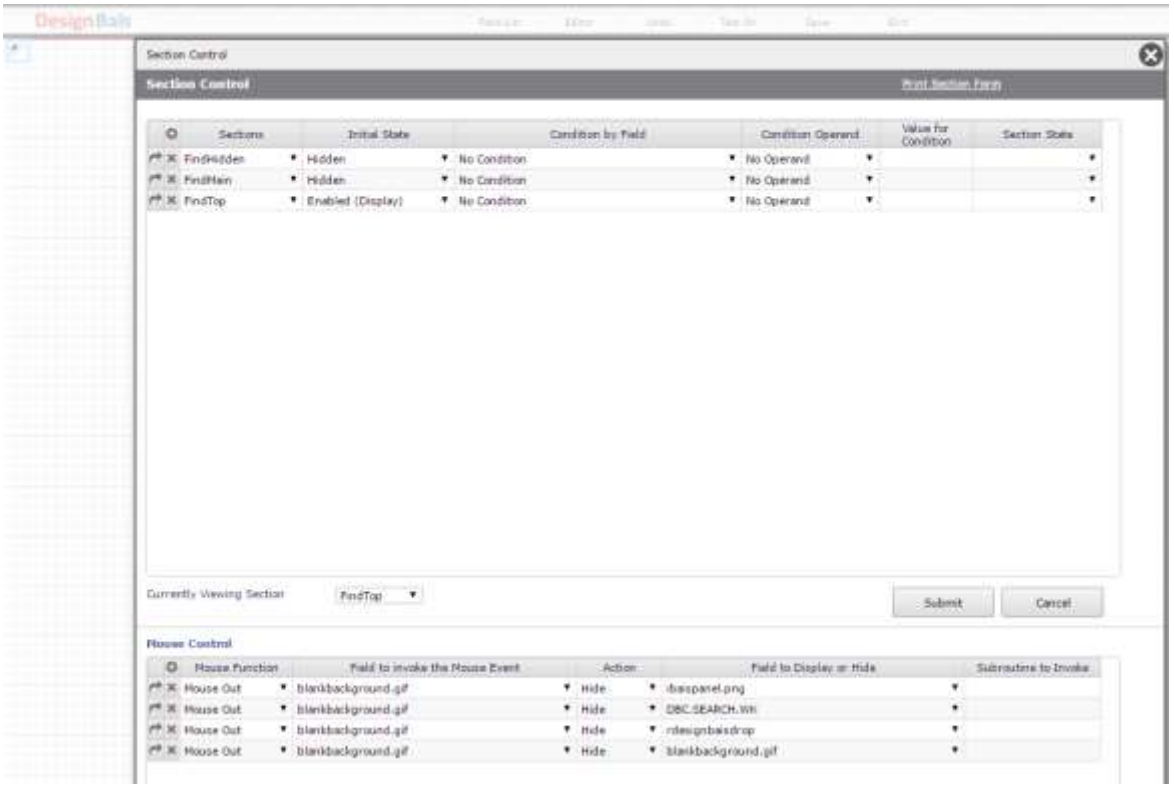
All fields on the header form will display, commencing in row 0, regardless of the header form depth, unless hidden or collapsed using section control or basic code. All main form fields will display from the row corresponding to the depth of the header form. If header form fields are positioned below the form depth of the header form they will display intermingled with fields of the main form. So if a header form has a form depth of zero all header fields will display intermingled with those on the main form.

By initially hiding all elements except one, on the header form, the developer can then allow a click or mouse over event, linked to the visible element of the header form, to display other parts of the header form. The overlay effect is only achieved by including a blank background image that covers the area of the main form that will be occupied by the header form in order to hide the main form fields that occupy the area occupied by the header form fields.

The following example demonstrates how to set this up. The example is available in the Demo Account in the menu item Predictive Text Demo Form (DBDEMO_PD).

The snip below shows that fields in two sections are hidden and the FindTop section is enabled. The field in FindTop is shown on the canvas as an arrow head image with a display class that invokes a contrasting hover image. Refer to the Field List snip below.

Section Control



Field List

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

You need to ensure that the z-index on the field properties of the header form are set to a number higher than the base form and within the header form fields will need to have the z-index set so that input fields display above the background images.

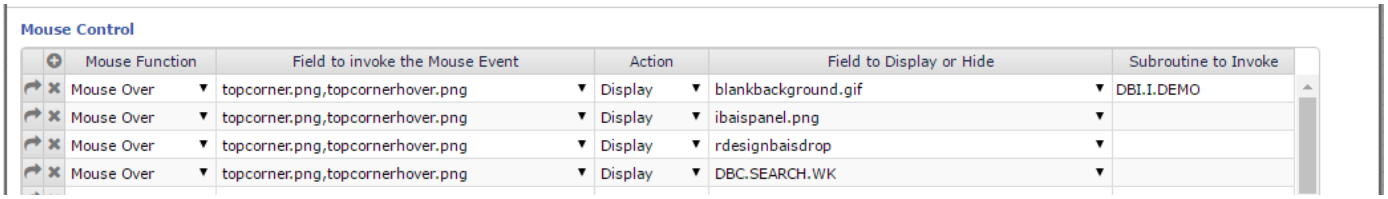
The Col and Row span of the fields must be set with two goals in mind. The “blankbackground.gif” must cover the area in which the header form will display, in order to cover the main forms fields that are under it. But it must also have an edge that extends past the “dbaispanel.png” image so that a mouseout event can be triggered, this being the method of closing the header form display.

You can invoke the display of the header form by either a click event on the image, in which case your basic code will have code that executes when the IMAGE event is encountered. This is the way that the demo form is set up.

The IMAGE code is as follows. This uses DBSECTIONSPEC to enable fields in the FindMain section and to set focus on the header field in which the user will enter a search string. (The third section FindHidden always remains hidden.)

```
*
IMAGE:
*
      BEGIN CASE
         CASE SCREEN.NO[1,2] = "PD" AND EVENTSOURCE = "topcorner.png,topcornerhover.png"
            * click on the header form image to enable the hidden header form fields
            SPECACTION = 'E'
            DBSECTIONSPEC<1,-1> = "FindMain"
            DBSECTIONSPEC<2,-1> = SPECACTION
            DBWORK<DBC.SEARCH.WK> = ""
            DBRETURN.TO.FIELD = 'DBC.SEARCH.WK'

      END CASE
      RETURN
```

Alternatively you can use the MOUSE OVER event in which case the display of fields will be triggered by entries loaded into the Mouse Control grid at the bottom of the Section Control form in Forms Designer.



Note that each field that is to be displayed must be entered. At least one line of the grid must contain the name of the subroutine to invoke so that the MOUSE OVER event can be processed in order to place focus on the field in which the user will enter a search string.

```
*
MOUSE.OVER:
*
      BEGIN CASE
         CASE EVENTSOURCE # ""
            DBWORK<DBC.SEARCH.WK> = ""
            DBRETURN.TO.FIELD = "DBC.SEARCH.WK"

      END CASE
      RETURN
```

In order to close the display of the header form the mouseout event is utilised. Refer to the Mouse Control section in the first snip above. Each element on the header form is hidden by a mouseout event associated with the the “blankbackground.gif” image.

In practice this is only required if the user clicks to open the header but then does not enter a search string, which would close the header form. The mouseout event will cause the header form to be closed if focus is placed outside of the “blankbackground.gif” image.

In most cases a VALIDATE event triggered by an entry in the search field on the header form will be intercepted by your basic subroutine. This code can, as in this example, pass the entered string as required. It them must set focus on the main form field and use DBSECTIONSPEC to hide the “FindMain” section of the header form:

```
*
```

```
VALIDATION:
*
         CASE SCREEN.NO = "PD" AND EVENTSOURCE = "DBC.FINDEX.KEY.WK"
            * the index DBCLIENT*1 defines this field to receive the selected key
            CLIENT.CODE = FIELD(DBVALUE,'|',2)
            DBPASS.DBVALUE = CLIENT.CODE
            DBPASS.DBVALUE.TO = "DBC.CLIENT.CODE"
            *
            DBRETURN.TO.FIELD = "DBC.CLIENT.NAME"
            DBSECTIONSPEC<1,-1> = "FindMain"
            DBSECTIONSPEC<2,-1> = "H"
            *
```

Chapter 30 – Creating I-Type Dictionaries

Creating I-Type Dictionaries

This is an example of setting up an I-Type dictionary for use in a selection.

The 'Client Class' on the DBEXEC file is calculated using the subroutine DB.D.DBEXEC.

One way to set this up in the 'Select Group Extract or Correlative' field is shown below.

The syntax for the 'SUBR()' function requires the name of the subroutine as the first argument, followed by other arguments.

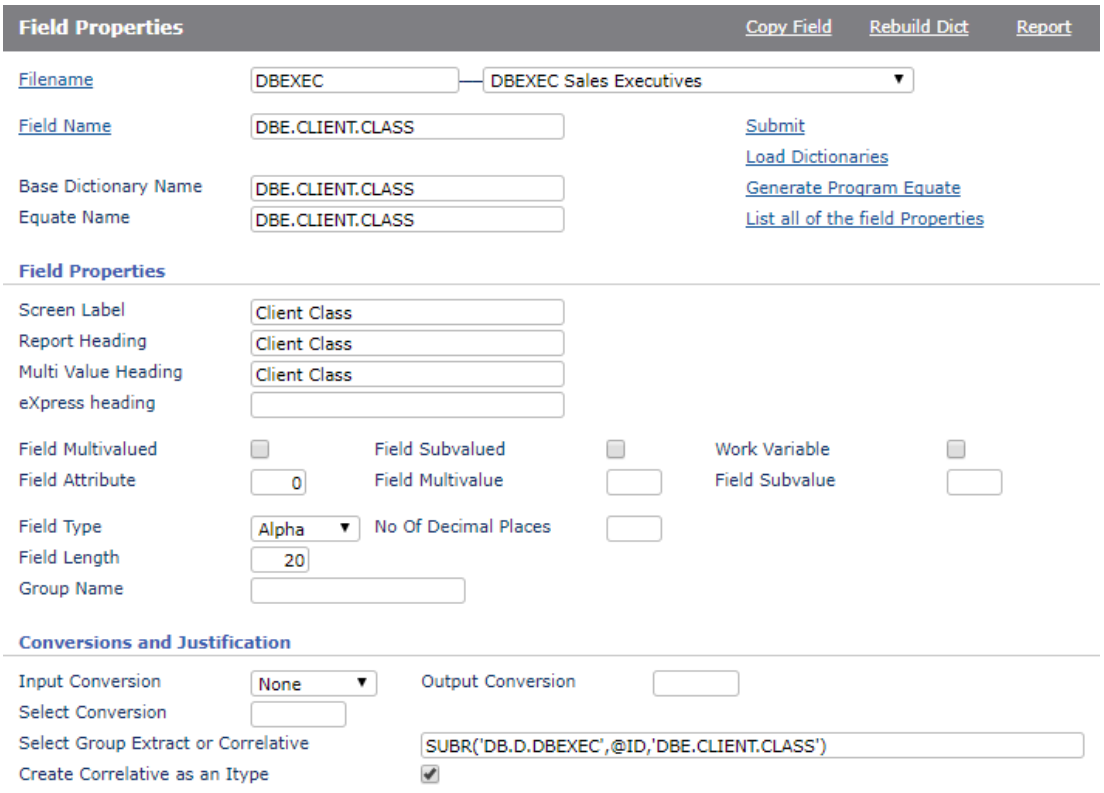
The subroutine itself must have the same number of arguments plus one. That extra one is the first argument which is used to return the value calculated by the subroutine.

In the example the Field Property has SUBR("DB.D.DBEXEC",@ID,'DBE.CLIENT.CLASS')

- @ID passes the record id of the record being processed
- 'DBE.CLIENT.CLASS' is surrounded by quotes and is therefore a string that the subroutine can test for
- If it is not in quotes then it must be the name of a dictionary in the DBEXEC file

SUBROUTINE DB.D.DBEXEC(RTNVAL,ID,NAME)

- RTNVAL as the first argument returns the value derived by the subroutine
- ID corresponds to @ID
- NAME is passed in with the value 'DBE.CLIENT.CLASS'



Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

```
SUBROUTINE DB.D.DBEXEC(RTNVAL,ID,NAME)
*
$INCLUDE DBI DBI.COMMON
$INCLUDE DBI DBI.SUB.COMMON
*
    BEGIN CASE
        CASE NAME = "DBE.CLIENT.CLASS"
            OPEN 'DBCLASS' TO F.DBCLASS THEN
                READ REC FROM F.DBCLASS,'C':ID THEN
                    RTNVAL = 'C':ID:' ':REC<1>
                END ELSE
                    RTNVAL = 'C':ID:' not defined'
                END
            END ELSE
                 RTNVAL = 'File DBCLASS missing'
            END
        CASE 1
            NULL
    END CASE
    RETURN
```

Listing the DBE.CLIENT.CLASS field on the DBEXEC file shows that where a code exists on the DBCLASS file that matches the pattern ‘C’ concatenated with the key of the DBEXEC file, then the value returned is the code followed by the name from attribute 1 (CCL.NAME).

```
SORT DBEXEC DBE.CLIENT.CLASS
DBEXEC.... Client Class.....

A          CA Class A
B          CB Class B
C          CC not defined
D          CD not defined
E          CE not defined
F          CF not defined
G          CG not defined
H          CH not defined
I          CI not defined
J          CJ not defined
K          CK not defined
L          CL not defined
M          CM not defined
N          CN not defined
O          CO not defined
P          CP Class P
Q          CQ Class Q
R          CR not defined
S          CS Class S
T          CT Class T
U          CU not defined
```

```
SORT DBCLASS = "C]" CCL.CODE CCL.NAME 05:24:08p
DBCLASS... Client Class Code Client Class Name.

C1          C1                  Class 1 C
C2          C2                  Class 2 C
CA          CA                  Class A
CB          CB                  Class B
CP          CP                  Class P
CQ          CQ                  Class Q
CS          CS                  Class S
CT          CT                  Class T
```

F-Correlative I-type Dictionaries

This example demonstrates how to create dictionaries to extract the first and last value from an attribute containing a list of multi-values. It is provided as an example of the method rather than as a dictionary that would necessarily be useful.

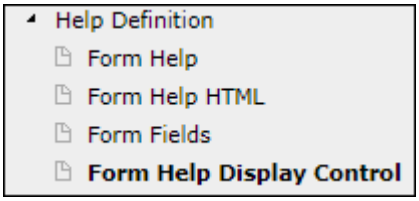
These two dictionaries reside in the DesignBais DBICHK file.

File Name	Field Property	Correlative
DBICHK	DBCK.FIRST.VAL	*EXTRACT(@RECORD,5,1,1)*
DBICHK	DBCK.LAST.VAL	EXTRACT(@RECORD,5,0,0);DCOUNT(@1,@VM);EXTRACT(@1,1,@2,0)

Chapter 31 – Form Help

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Form Help



The Help Definition side menu options allow you to select from two form help maintenance forms. The Form Help option creates DBIHELP records that display an on-form report listing form fields and field help. The newer Form Help HTML option displays a form that includes an iframe to allow the display of html Form Help Text.

The Form Help Display Control option allows you to select which form help to display. It also allows the option for users in the Developers user group to see a link to the field properties form for each field on a form.

Form Help is invoked by placing a button called B.DBIFORMHELP on your form. This button must call the subroutine DBI.G.BUTTON from the Process After slot.

To emulate the help button at the top right of most DesignBais tool forms place the following string in the Field Text of the button:

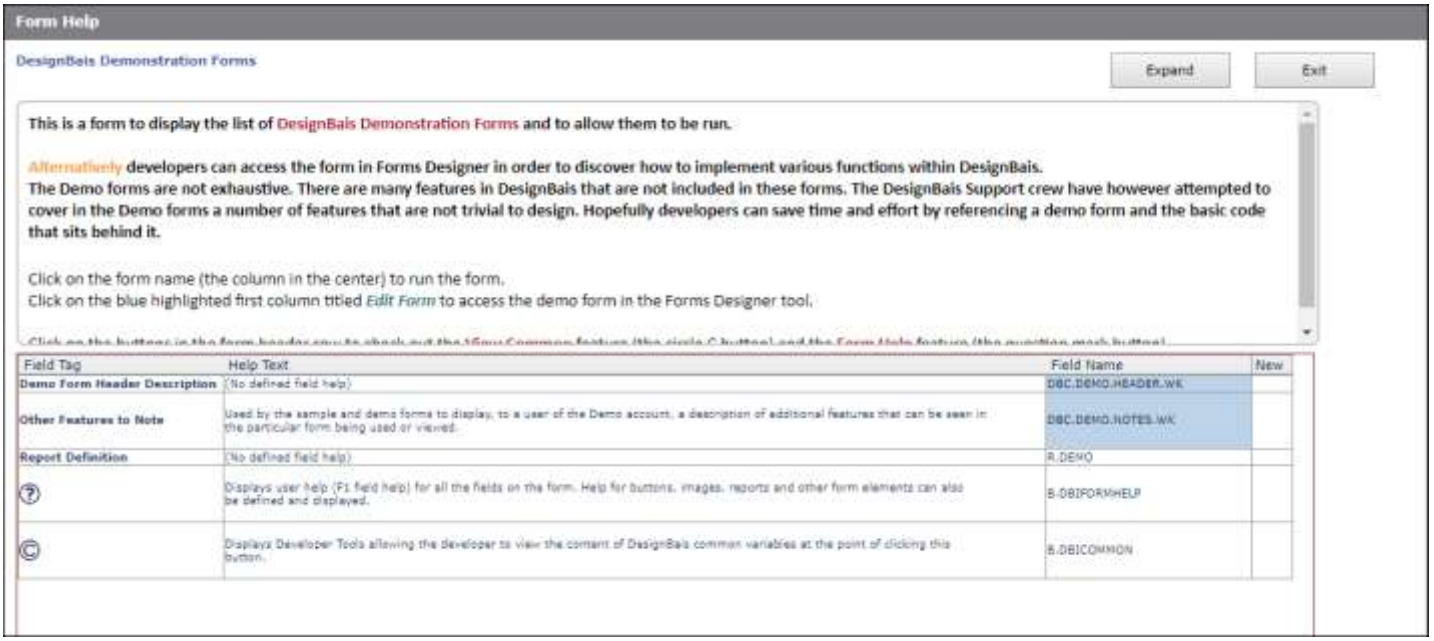
**

If you are placing the button on a label header text field use the display Class "*dbaisSearchLabelLite"* otherwise use "*dbaisSearchLabelBlue*".

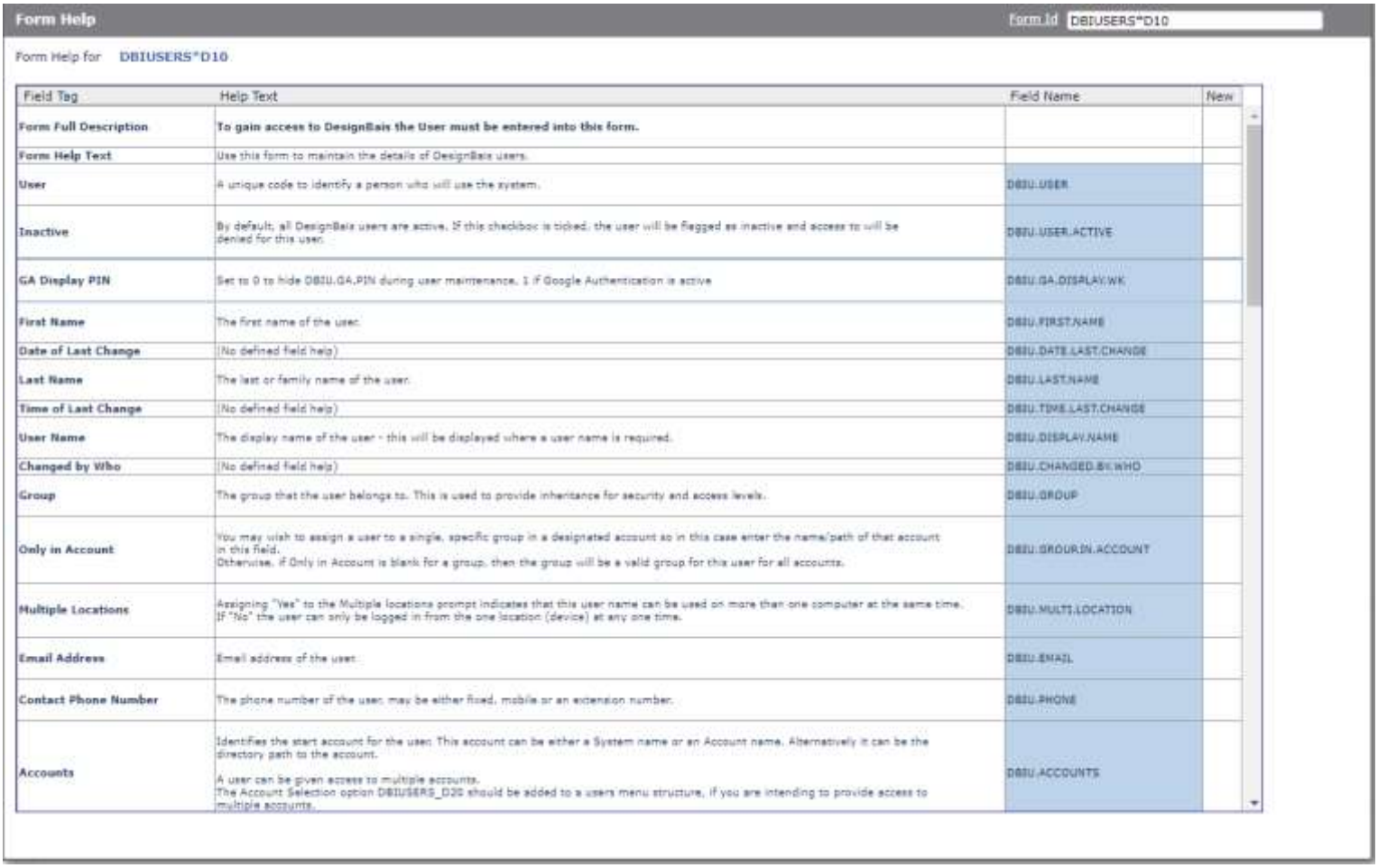
The *Add Button* option on the *Code Editor* form automates this procedure. It allows you to add or remove the form help button from forms.

Form Help displays all, or selected, fields from the form from which it is called showing the Field Tag, the F1 Help text and the Field Name. Users who are in the Developers User Group are able to click highlighted field names in order to display the Field Properties form for the selected field, provided the field properties are on the DBIPROP file. Field properties from DBISYSPROP will not be highlighted and cannot be accessed by clicking on the Field Name column.

The new HTML Form Help display is shown here.



The original Form Help display is shown here.



Form Help

Form Id DBIUSERS"D10

Form Help for DBIUSERS"D10

Field Tag	Help Text	Field Name	New
Form Full Description	**To gain access to DesignBais the User must be entered into this form.**		
Form Help Text	Use this form to maintain the details of DesignBais users.		
User	A unique code to identify a person who will use the system.	DBIU.USER	
Inactive	By default, all DesignBais users are active. If this checkbox is ticked, the user will be flagged as inactive and access to will be denied for this user.	DBIU.USER.ACTIVE	
GA Display PIN	Set to 0 to hide DBIU.GA.PIN during user maintenance. 1 if Google Authentication is active	DBIU.GA.DISPLAY.WK	
First Name	The first name of the user.	DBIU.FIRST.NAME	
Date of Last Change	(No defined field help)	DBIU.DATE.LAST.CHANGE	
Last Name	The last or family name of the user.	DBIU.LAST.NAME	
Time of Last Change	(No defined field help)	DBIU.TIME.LAST.CHANGE	
User Name	The display name of the user - this will be displayed where a user name is required.	DBIU.DISPLAY.NAME	
Changed by Who	(No defined field help)	DBIU.CHANGED.BY.WHO	
Group	The group that the user belongs to. This is used to provide inheritance for security and access levels.	DBIU.GROUP	
Only in Account	You may wish to assign a user to a single, specific group in a designated account so in this case enter the name/path of that account in this field. Otherwise, if Only in Account is blank for a group, then the group will be a valid group for this user for all accounts.	DBIU.GROUP.IN.ACCOUNT	
Multiple Locations	Assigning "Yes" to the Multiple locations prompt indicates that this user name can be used on more than one computer at the same time. If "No" the user can only be logged in from the one location (device) at any one time.	DBIU.MULTI.LOCATION	
Email Address	Email address of the user.	DBIU.EMAIL	
Contact Phone Number	The phone number of the user, may be either fixed, mobile or an extension number.	DBIU.PHONE	
Accounts	Identifies the start account for the user. This account can be either a System name or an Account name. Alternatively it can be the directory path to the account. A user can be given access to multiple accounts. The Account Selection option DBIUSERS_D20 should be added to a users menu structure, if you are intending to provide access to multiple accounts.	DBIU.ACCOUNTS	

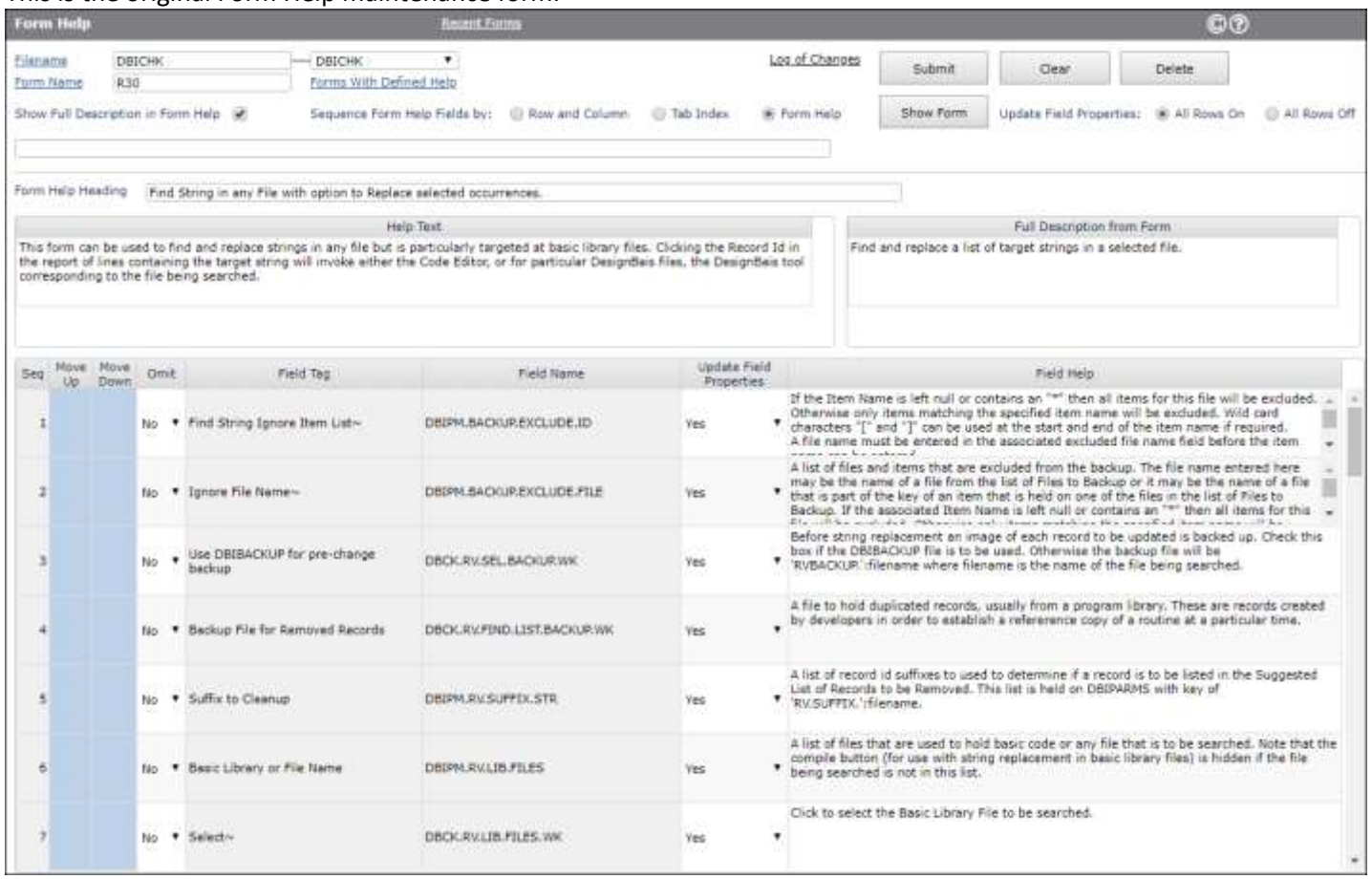
The options on the *Help* Definition side menu called *Form Help* and *Form Help HTML* are used to create form help.

These forms allow you to display and maintain F1 Help text for all fields on a form. In addition you can control which fields display in Form Help, you can enter more detailed *Form Help Text*, you can change the sequence of fields in the display and you can enter help for buttons and reports. Submit this form to create a form help record on the DBIHELP file. If a record exists for a form then this record is retrieved when the help button on a form is clicked. In this case any fields that have been added to the form since the creation of the help definition form will be shown at the end of the help form with an asterisk in the *New* column.

The HTML option includes a field that allows you to create Form Help Text using the HTML Editor. The resulting display can utilize the full suite of HTML options.

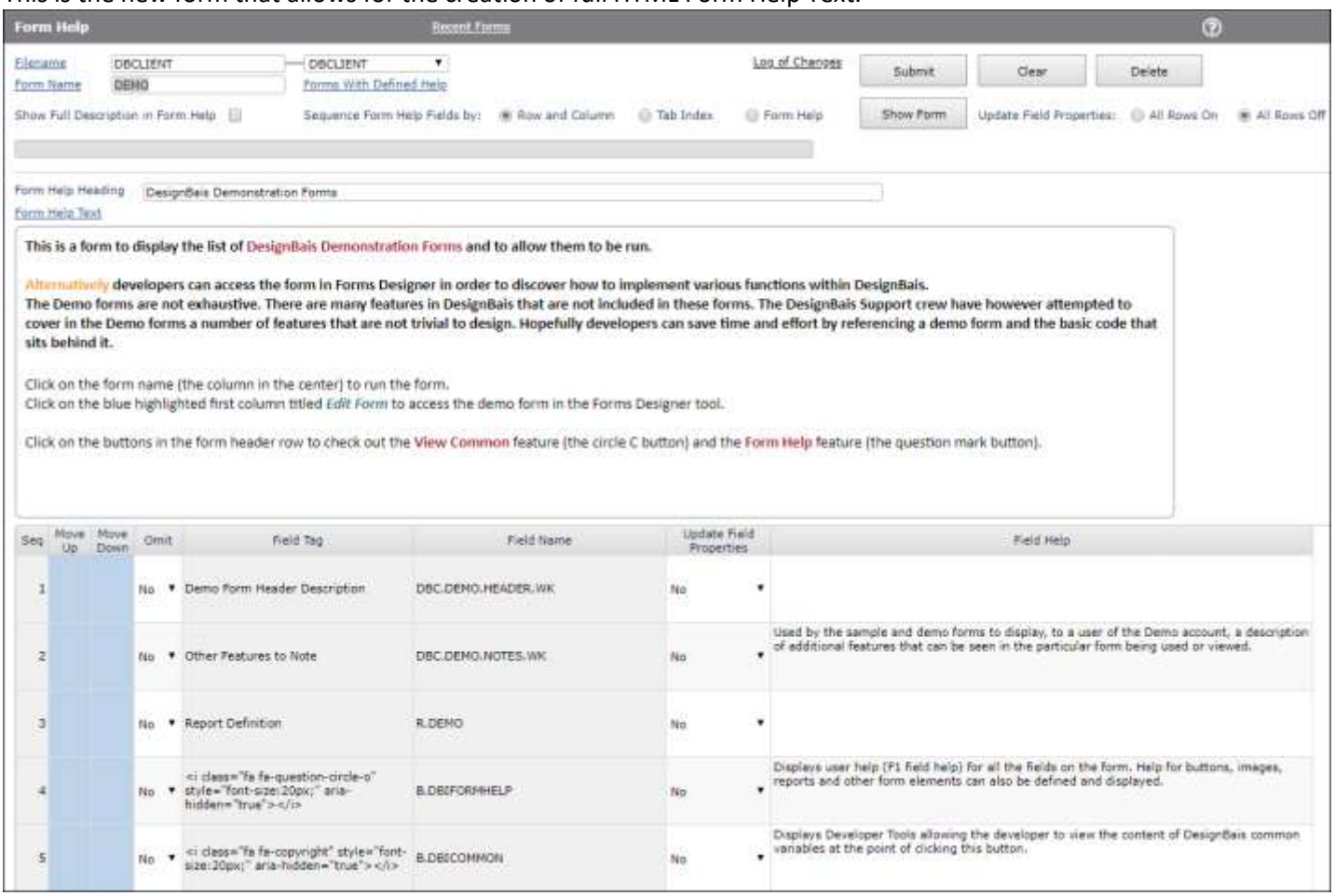
If there is no help definition form saved on DBIHELP then form help is built on the fly by reading the form record so in effect the Form Help can display without any additional effort. You do not have to create the help definition.

This is the original Form Help maintenance form.



Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

This is the new form that allows for the creation of full HTML Form Help Text.



Prompts

Filename — Enter the name of a DesignBais File from your DBIFILES file, or select from the dropdown list. Alternatively click the Filename text which is a hyperlink.

Form Name — Enter the name of the form for which form help is to be set up. Alternatively click the Form Name text which is a hyperlink and select a form.

Forms With Defined Help

This hyperlink will display a list of the forms which have Form Help already set up. The Form Help records are held on the DBIHELP file with a type (DBIH.TYPE attribute 31) of “FORMHELP”. This is to distinguish these type of records from other help type records that may exist on the DBIHELP file.

Show Full Description in Form Help

Clicking this check box causes the Full Description from the Form to be displayed in the Form Help.

Sequence Form Help Fields by

Sets the sequence in which the fields on the form are listed on the Form Help.
Row and Column and *Tab Index* sort the fields based on the sequence of fields on the form record. The *Form Help* option allows you to set the sequence you want within this Form Help set up form. Use the *Move Up* and *Move Down* columns in the display to re-arrange the fields.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

After submitting this form the Form Help display form FHD will display the fields on the form in the defined sequence.

Note that moving fields up or down automatically sets this field to the *Form Help* setting.

Update Field Properties — Sets Click *All Rows On* to toggle the grid column *Update Field Properties* to *Yes* for all grid rows. Similarly click *All Rows Off* to toggle the grid column *Update Field Properties* to *No*.

Form Help Heading — Enter text to appear at the top of the Form Help after the "Form Help for " tag.

Help Text — Enter text to appear in the Form Help Text row of the Form Help. This text can be used to provide additional help for using a form or to describe any other aspect that will assist the users.

Fields in the Grid

Omit — Select *No* or *Yes* from the dropdown. Selecting *Yes* allows fields to be omitted from the Form Help display.

Move Up / Down — Click a row to move the help text for a field either up or down the list. This allows the developer to display the field help in a sequence different to the row and column or tab sequence of the form.

Update Field Properties — This setting controls whether the help text displayed here is to update the Field Properties *Help Text* field. Normally this setting should be set to *Yes* so that field help can be maintained here and then updated to the field property record. Set to *No* if form specific help is required. When running a form the F1 key will display field help from the help form if help for the field exists on the help form, otherwise the help text from the field property record is displayed.

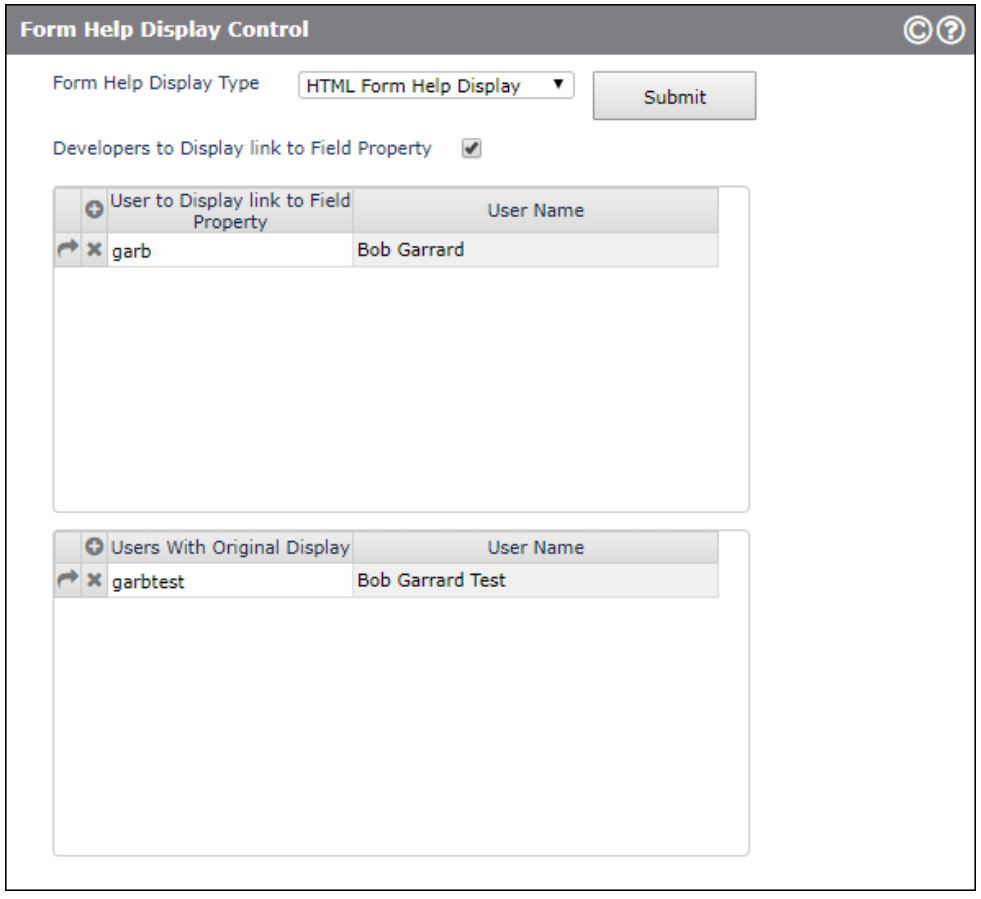
Field Help — This is the F1 field help from the Field Properties record for the field. Any changes that you make here will be updated to the Field Properties record when the Submit button is clicked, unless *Update Field Properties* for the row is set to *No*. Form specific field help is therefore possible.

Form elements that do not have a record on the Field Properties file, such as buttons, images and reports will initially have no text in this Field Help column. Text entered here, however, is saved on the DBIHELP record and will then appear on the Form Help display.

Buttons	
Submit	Will update the DBIHELP record with Id Filename* Form Name.
Clear	Will clear the form and discard any changes.
Delete	Will delete the form from the DBIHELP file.
Show Form	Will display the form that is being maintained in the Form Help form. This allows you to see the actual form that is being documented.
Log of Changes	Displays an audit trail of the fields that have had the F1 help text amended in this help record.

Form Help Display Control

Use this form to select which form is to be used for Form Help display.



Check the *Developers to Display link to Field Property* if you wish to activate this option.

Alternatively individual user ids can be entered so that the link is activated regardless of the setting of the Developers check box.

You can alos allow users to see the older Form Help display form by entering users ids in the *Users Wih Original Display* field.

Display Common Variables

Placing a button called *B.DBICOMMON* on a form, with a Process After of *DBI.G.BUTTON* enables the developer to display the values held in DesignBais *DBI.COMMON* variables.

The *Form Help* form can be used to put the button on the form that is being maintained. Refer to the hyperlink *Add Display Common Button to Form*.

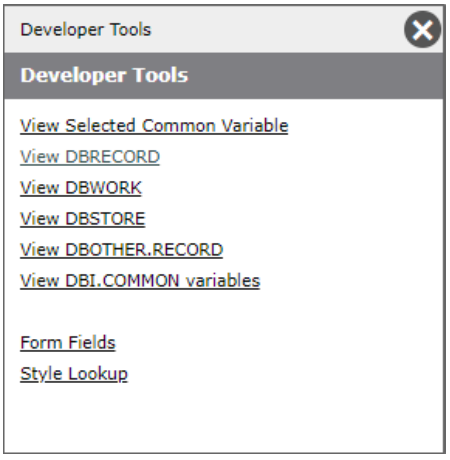
The standard Display Common button is a *C* inside a circle (the copyright symbol).

The ability to display the contents of common variables can be very useful during devlopment to verify that data is populated as intended.

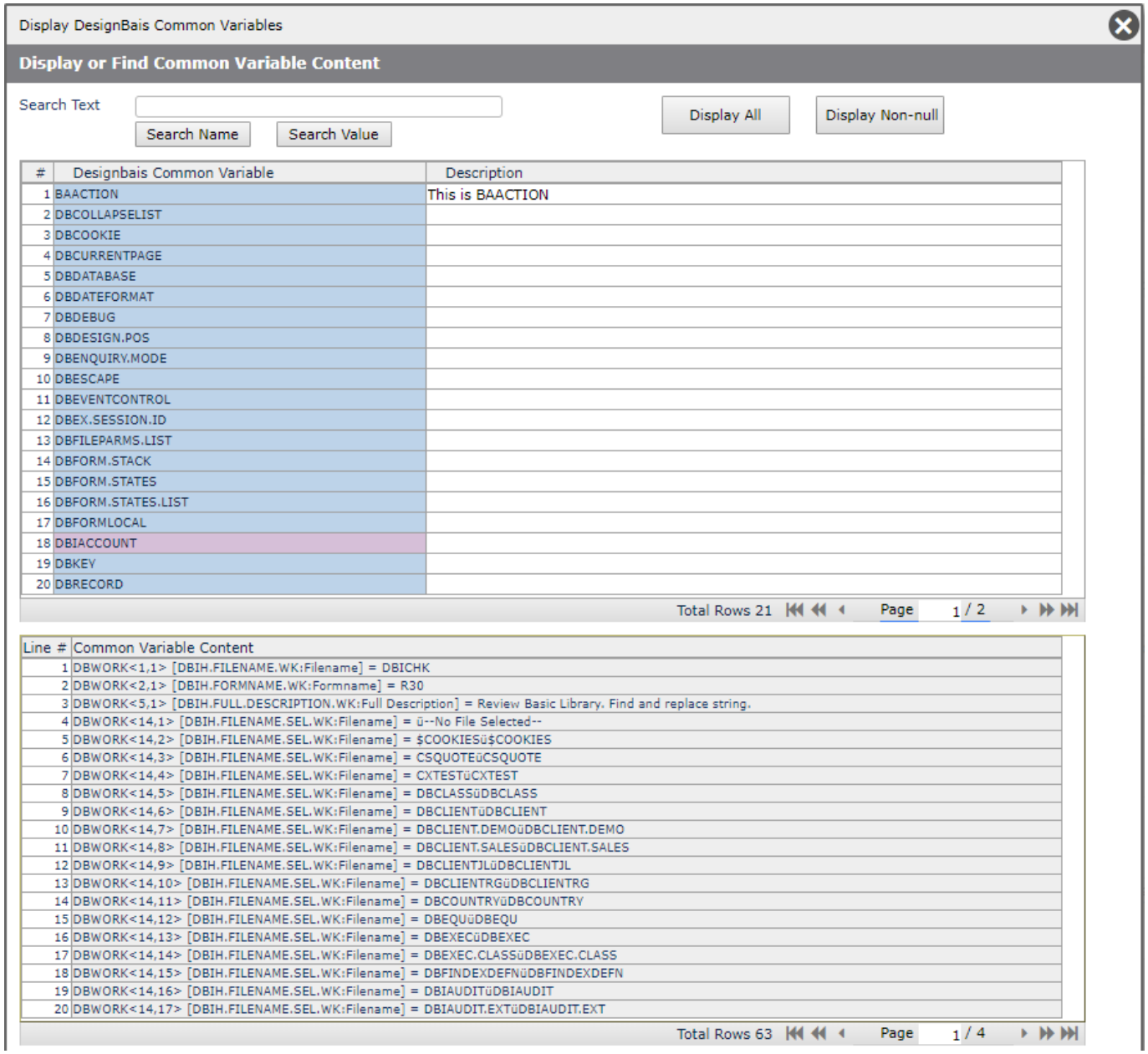
When the Display Common button is clicked a record of the contents of DesignBais common variables is written to records in the DBIPARMS file with ids like *COM*userid*variablename*. See example below:

COM*garb*DBCOMMON
COM*garb*DBOTHER.RECORD
COM*garb*DBRECORD
COM*garb*DBSTORE
COM*garb*DBWORK
COM*garb*SCREENROOT
COM*legj*DBCOMMON
COM*legj*DBOTHER.RECORD
COM*legj*DBRECORD
COM*legj*DBWORK
COM*legj*SCREENROOT
COM.COMMON.VAR.LIST
COM.VAR.LIST

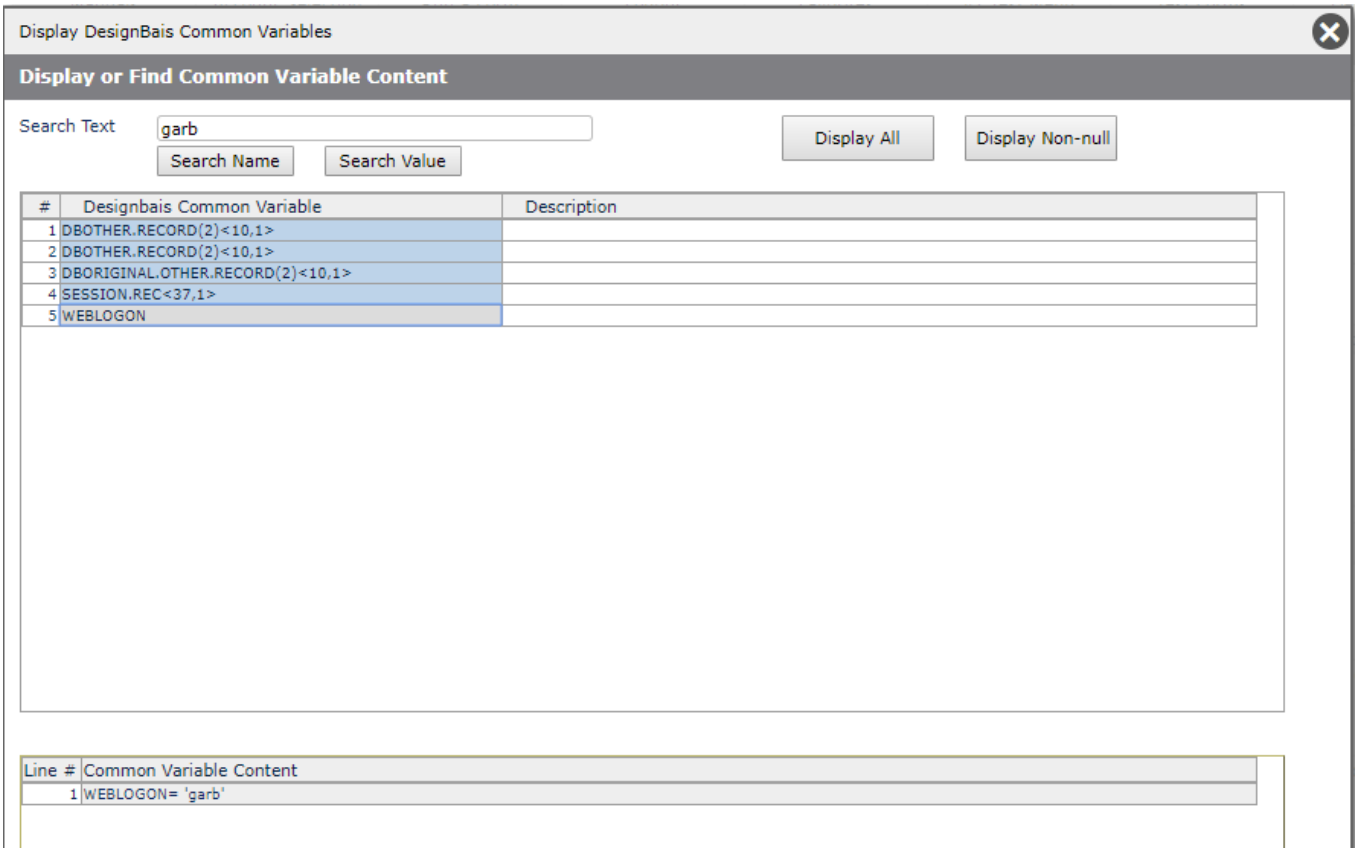
These records can then be displayed in On-form Reports by clicking hyperlinks on the *Developer Tools* form:



The *View Selected Common Variable* form, see below, also allows the developer to enter a text string in the *Search Text* field and to search for either a particular common variable name containing the text entered, or for the common variables that contain the text entered.



The *Description* column in the above form is an input field. Developers can describe common variables, or add short notes about particular variables, if this is helpful. The descriptions are not multi-user, there is one record per account to hold the descriptions.

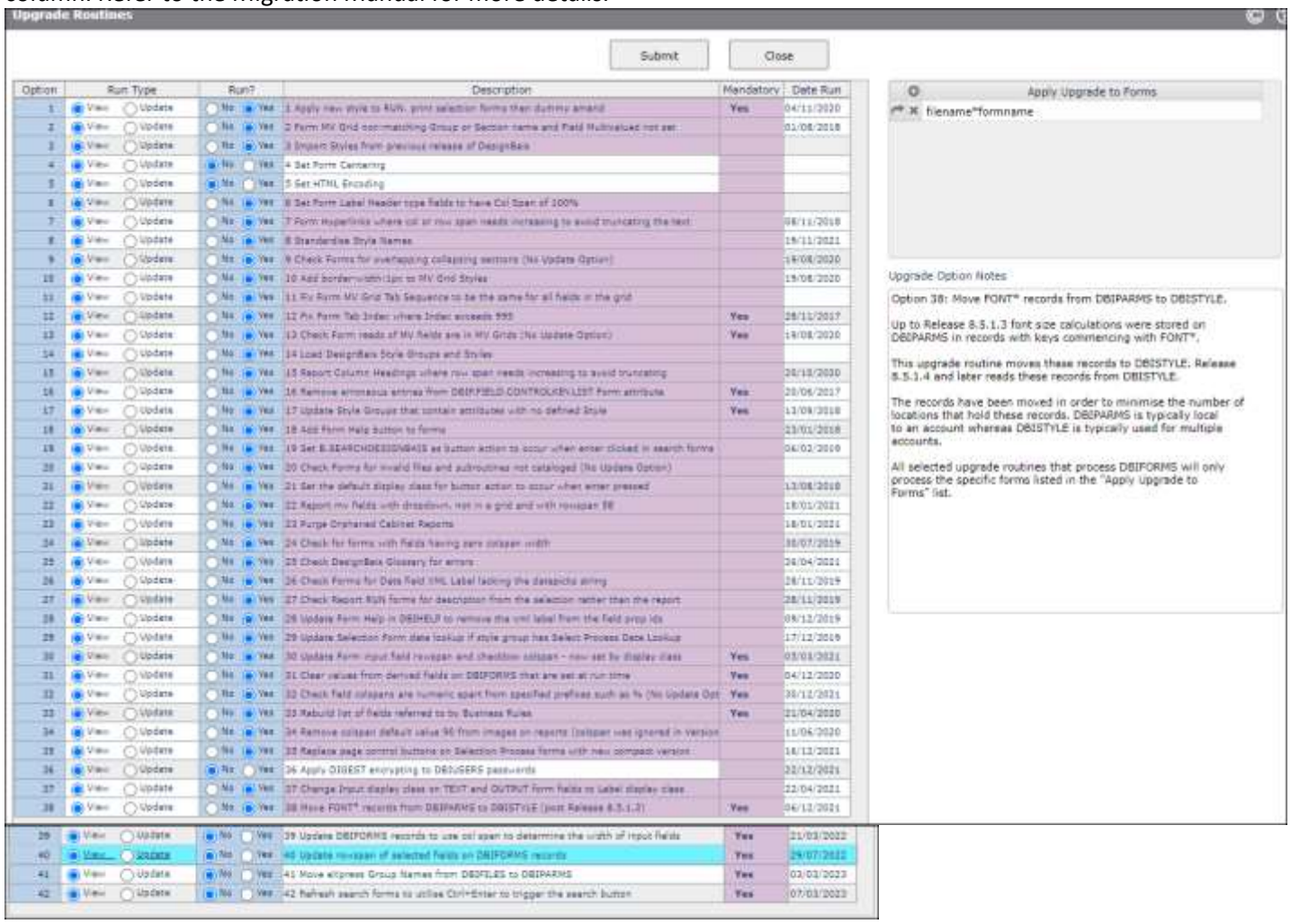


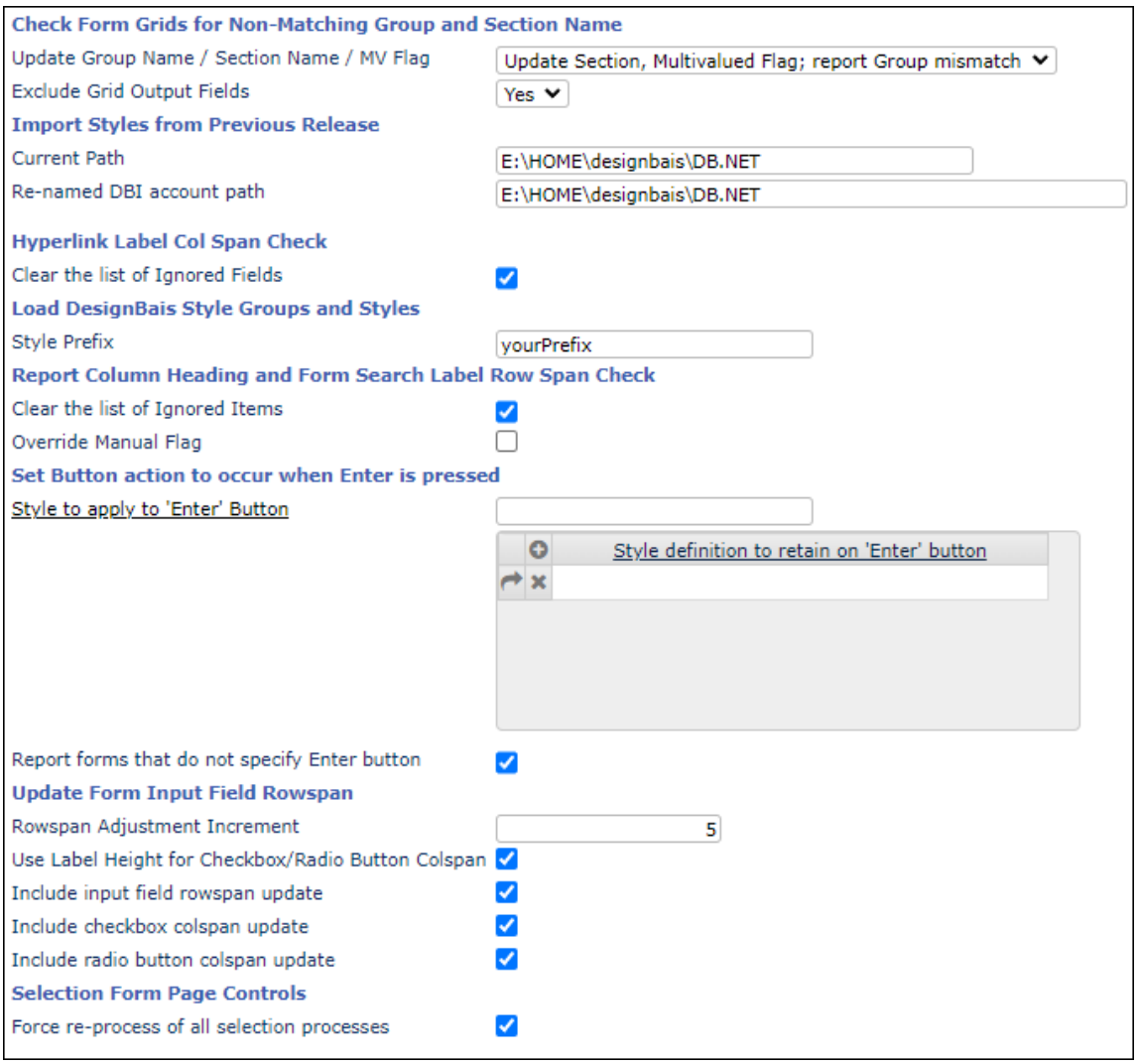
In practice it may be helpful to have the Display Common button on a form during development and then to remove the button prior to moving the form to production. The *Remove Display Common button* on the Form Help form allows the button to be removed without having to go into the Forms Designer.

Chapter 32 – Upgrade/Migration

Upgrade / Migration Tools

The Upgrade Routines as at this release are listed. Options can be selected to run in either ‘View’ or ‘Update’ mode. Multiple options can be selected for a single run. Mouseover help is displayed by hovering over the Description column. Refer to the Migration Manual for more details.





Options flagged as ‘Mandatory’ must be run in ‘Update’ mode in Version 7 and above.

The Date Run is updated when an option is executed in ‘Update’ mode.

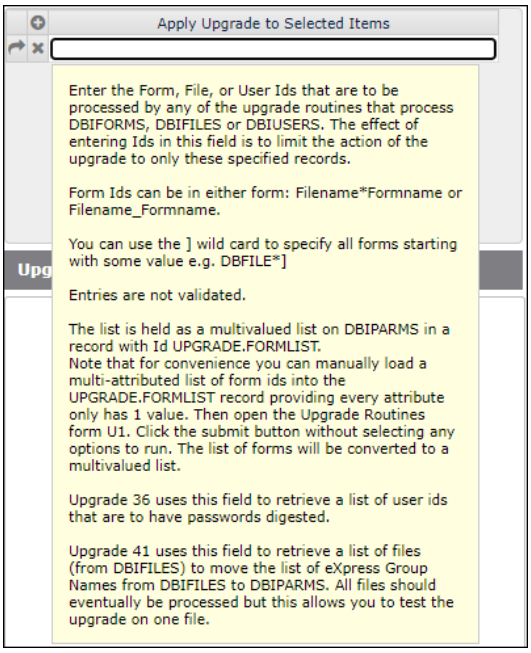
Use the Apply Upgrade to Selected Items grid to enter a list of forms that are to be processed by the upgrade routiens. In this way you can limit the number of forms that are processed.

The list is held as a multivalue list in attribute 1 of the DBIPARMS record with Id UPGRADE.FORMLIST.

In order to faciliate the entry of a large number of forms you can select forms using the command line in a telnet session. Save the list. Then use COPY-LIST to copy the list to DBIPARMS UPGRADE.FORMLIST. This creates a multi-attribute list in the record but the upgrade routine will convert this to a multivalue list after the *Submit* button is clicked. Note that your database may limit the number of forms that can be selected from this list.

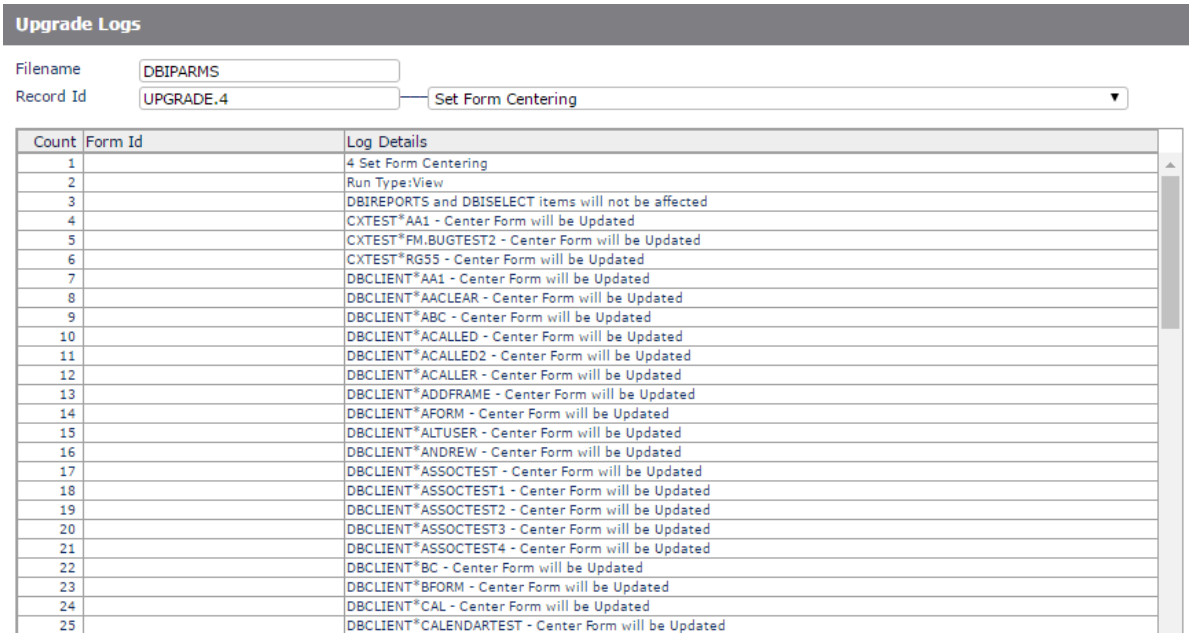
The select is of the form:

SSELECT DBIFORMS
"DBDEMO*ADDFRAME""DBDEMO*AWESOME""DBDEMO*BOBCAROUSEL""DBDEMO*CAB""DBDEMO*CALENDAR""DBDEMO*CAPTCHA""DBDEMO*CAROUSEL""DBDEMO*CHART""DBDEMO*CLICKAFTER""DBDEMO*D80""DBDEMO*DAYPICKER""DBDEMO*DBCALLURL""DBDEMO*DBHB""DBDEMO*DBHBBC""DBDEMO*DBMAIL"



The Upgrade Logs option displays the results of the upgrade routines. These logs are held on the DBIPARMS file with an id of 'UPGRADE.n' where 'n' is the option number.

In the example below for option 4 the log lists the forms that will be updated if option 4 (Set Form Centering) is executed in update mode.



Upgrade Logs

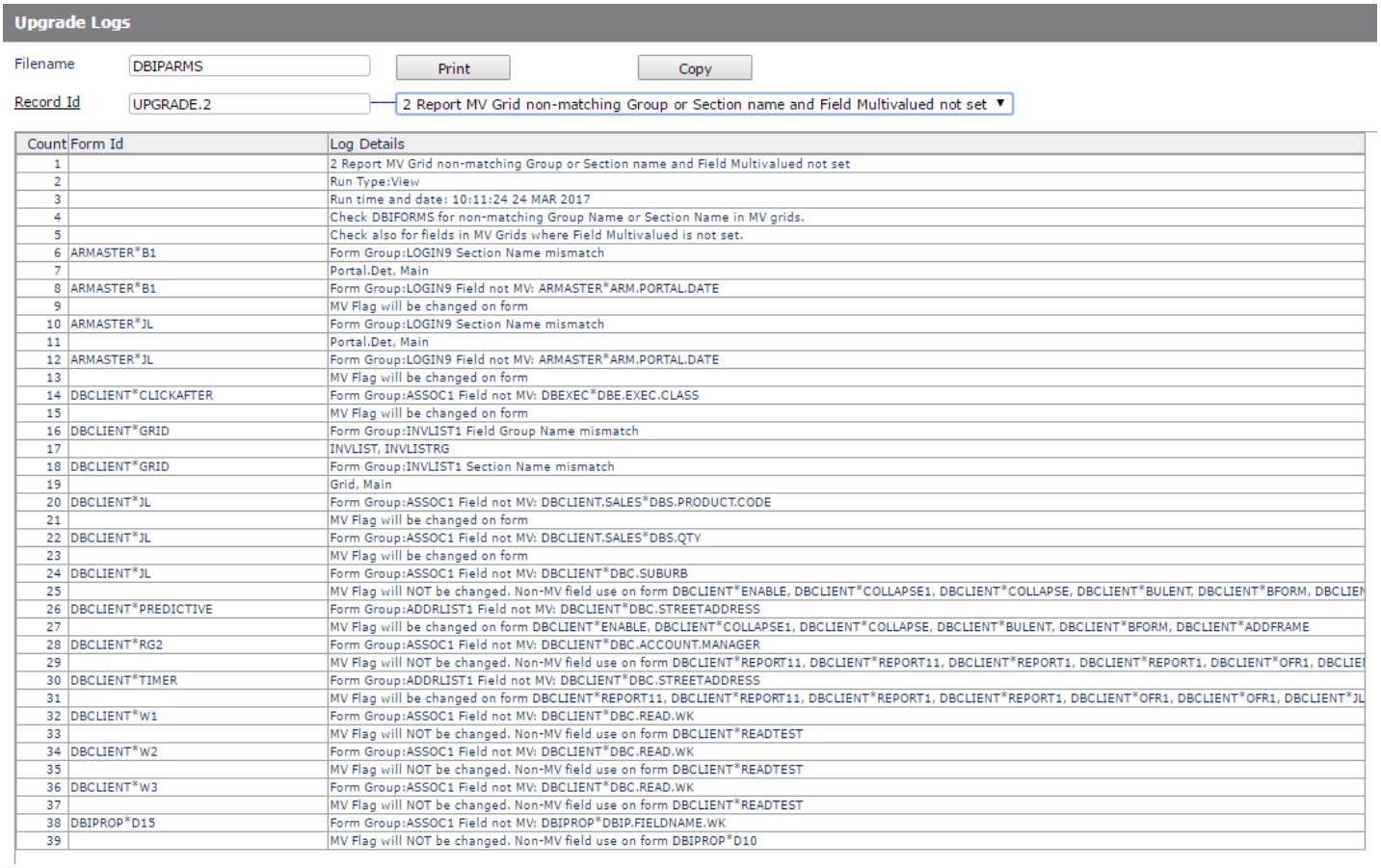
Filename: DBIPARMS

Record Id: UPGRADE.4 — Set Form Centering

Count	Form Id	Log Details
1		4 Set Form Centering
2		Run Type:View
3		DBIREPORTS and DBISELECT items will not be affected
4		CXTEST*AA1 - Center Form will be Updated
5		CXTEST*FM.BUGTEST2 - Center Form will be Updated
6		CXTEST*RG55 - Center Form will be Updated
7		DBCLIENT*AA1 - Center Form will be Updated
8		DBCLIENT*AACLEAR - Center Form will be Updated
9		DBCLIENT*ABC - Center Form will be Updated
10		DBCLIENT*ACALLED - Center Form will be Updated
11		DBCLIENT*ACALLED2 - Center Form will be Updated
12		DBCLIENT*ACALLER - Center Form will be Updated
13		DBCLIENT*ADDFRAME - Center Form will be Updated
14		DBCLIENT*AFORM - Center Form will be Updated
15		DBCLIENT*ALTUSER - Center Form will be Updated
16		DBCLIENT*ANDREW - Center Form will be Updated
17		DBCLIENT*ASSOCTEST - Center Form will be Updated
18		DBCLIENT*ASSOCTEST1 - Center Form will be Updated
19		DBCLIENT*ASSOCTEST2 - Center Form will be Updated
20		DBCLIENT*ASSOCTEST3 - Center Form will be Updated
21		DBCLIENT*ASSOCTEST4 - Center Form will be Updated
22		DBCLIENT*BC - Center Form will be Updated
23		DBCLIENT*BFORM - Center Form will be Updated
24		DBCLIENT*CAL - Center Form will be Updated
25		DBCLIENT*CALENDARTEST - Center Form will be Updated

Option 2 checks forms for the use of fields within multivalue grids, where those fields are not flagged as multivalued. In update mode these fields will be amended to set the 'Field Multivalued' property on. The exception to this is where a field is set for both single value and multivalue use. In this case the log will report this. Notice particularly the final entry on the log for Form DBIPROP*D15 where field DBIP.FIELDNAME.WK is reported as being in a multivalue grid on form DBIPROP*D15 and used as a single value on DBIPROP*D10. Note that Option 2 writes temporary records to the DBICON file. It is recommended that this file is sized to have a modulo greater than the twice the number of form records in the DBIFORMS file.

In this scenario the developer will have to review the use of the field and amend the forms manually. The normal way to resolve the problem is to create a new field, flagged as multivalued, and use this in the multivalue grid.



Upgrade Logs

Filename DBIPARMS Print Copy

Record Id UPGRADE.2 2 Report MV Grid non-matching Group or Section name and Field Multivalued not set

Count	Form Id	Log Details
1		2 Report MV Grid non-matching Group or Section name and Field Multivalued not set
2		Run Type:View
3		Run time and date: 10:11:24 24 MAR 2017
4		Check DBIFORMS for non-matching Group Name or Section Name in MV grids.
5		Check also for fields in MV Grids where Field Multivalued is not set.
6	ARMASTER*B1	Form Group:LOGIN9 Section Name mismatch
7		Portal.Det, Main
8	ARMASTER*B1	Form Group:LOGIN9 Field not MV: ARMASTER*ARM.PORTAL.DATE
9		MV Flag will be changed on form
10	ARMASTER*JL	Form Group:LOGIN9 Section Name mismatch
11		Portal.Det, Main
12	ARMASTER*JL	Form Group:LOGIN9 Field not MV: ARMASTER*ARM.PORTAL.DATE
13		MV Flag will be changed on form
14	DBCLIENT*CLICKAFTER	Form Group:ASSOC1 Field not MV: DBEXEC*DBE.EXEC.CLASS
15		MV Flag will be changed on form
16	DBCLIENT*GRID	Form Group:INVLIST1 Field Group Name mismatch
17		INVLIST, INVLISTRG
18	DBCLIENT*GRID	Form Group:INVLIST1 Section Name mismatch
19		Grid, Main
20	DBCLIENT*JL	Form Group:ASSOC1 Field not MV: DBCLIENT.SALES*DBS.PRODUCT.CODE
21		MV Flag will be changed on form
22	DBCLIENT*JL	Form Group:ASSOC1 Field not MV: DBCLIENT.SALES*DBS.QTY
23		MV Flag will be changed on form
24	DBCLIENT*JL	Form Group:ASSOC1 Field not MV: DBCLIENT*DBC.SUBURB
25		MV Flag will NOT be changed. Non-MV field use on form DBCLIENT*ENABLE, DBCLIENT*COLLAPSE1, DBCLIENT*COLLAPSE, DBCLIENT*BULENT, DBCLIENT*BFORM, DBCLIEN
26	DBCLIENT*PREDICTIVE	Form Group:ADDRLIST1 Field not MV: DBCLIENT*DBC.STREETADDRESS
27		MV Flag will be changed on form DBCLIENT*ENABLE, DBCLIENT*COLLAPSE1, DBCLIENT*COLLAPSE, DBCLIENT*BULENT, DBCLIENT*BFORM, DBCLIENT*ADDFRAME
28	DBCLIENT*RG2	Form Group:ASSOC1 Field not MV: DBCLIENT*DBC.ACCOUNT.MANAGER
29		MV Flag will NOT be changed. Non-MV field use on form DBCLIENT*REPORT11, DBCLIENT*REPORT11, DBCLIENT*REPORT1, DBCLIENT*REPORT1, DBCLIENT*OFR1, DBCLIE
30	DBCLIENT*TIMER	Form Group:ADDRLIST1 Field not MV: DBCLIENT*DBC.STREETADDRESS
31		MV Flag will be changed on form DBCLIENT*REPORT11, DBCLIENT*REPORT11, DBCLIENT*REPORT1, DBCLIENT*REPORT1, DBCLIENT*OFR1, DBCLIENT*OFR1, DBCLIENT*JL
32	DBCLIENT*W1	Form Group:ASSOC1 Field not MV: DBCLIENT*DBC.READ.WK
33		MV Flag will NOT be changed. Non-MV field use on form DBCLIENT*READTEST
34	DBCLIENT*W2	Form Group:ASSOC1 Field not MV: DBCLIENT*DBC.READ.WK
35		MV Flag will NOT be changed. Non-MV field use on form DBCLIENT*READTEST
36	DBCLIENT*W3	Form Group:ASSOC1 Field not MV: DBCLIENT*DBC.READ.WK
37		MV Flag will NOT be changed. Non-MV field use on form DBCLIENT*READTEST
38	DBIPROP*D15	Form Group:ASSOC1 Field not MV: DBIPROP*DBIP.FIELDNAME.WK
39		MV Flag will NOT be changed. Non-MV field use on form DBIPROP*D10

There is a Print option to generate a printed copy of the log.

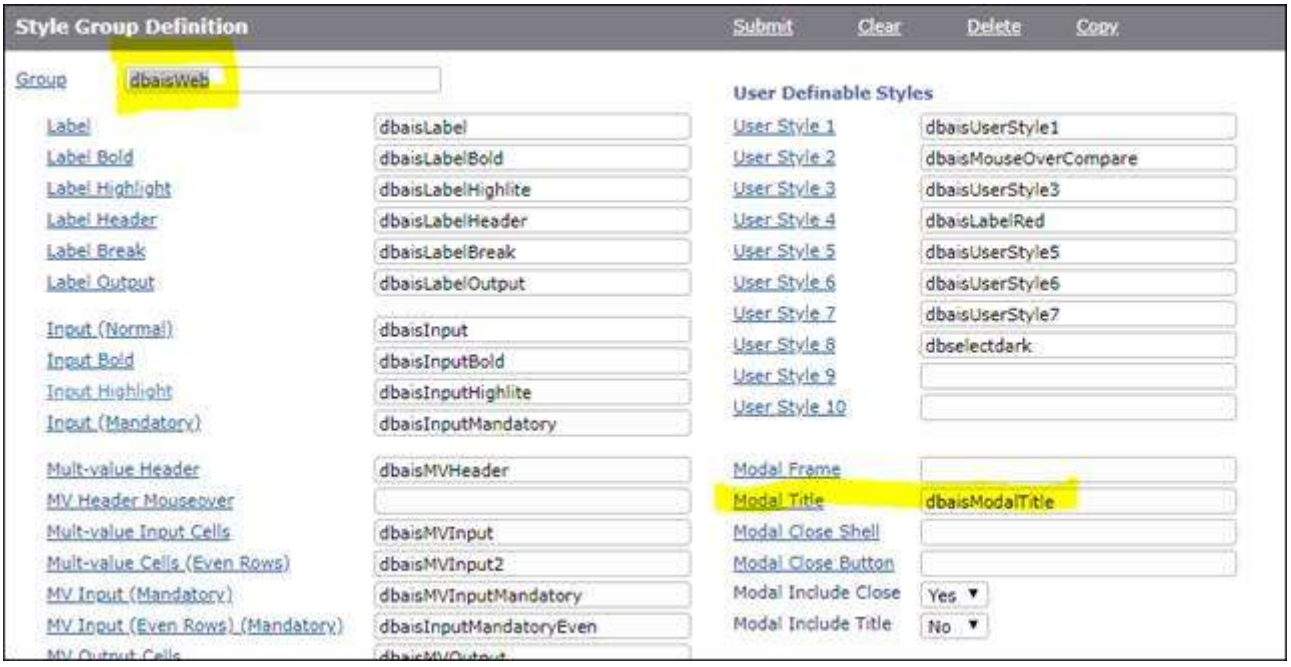
There is a Copy button. This allows you to create a copy of a log record. A naming convention is enforced. The key of a log record as generated is ‘UPGRADE.n’ where ‘n’ is the Option number of the upgrade routine. Log records can only be copied to a new id commencing with ‘UPGRADE.n.’ thus maintaining a link to the original option.

The log record will be overwritten each time an option is executed so the Copy function is useful to retain a record of changes that were done in previous runs.

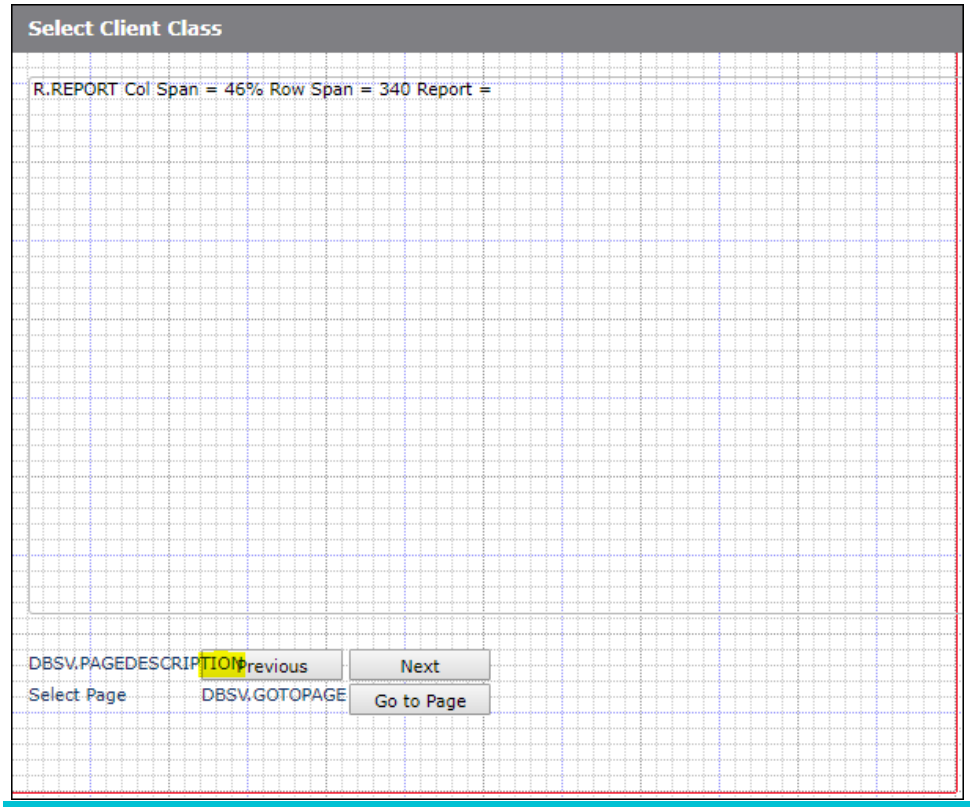
The Form Id in each log report is a hyperlink to allow you to review the form in Forms Designer.

Option 17 is useful to ensure your style groups have a class defined for all required slots. In Version 6 some slots in the style group were not required. In Release 7/8 this may have changed. In the example below The *Modal Title* slot was usually not set in Version 6. It is needed in Release 7/8 to ensure, among other things, that modal form title bars are not transparent.

Running option 17 will place the style *dbaisModalTitle* (from the standard *dbaisWeb* style group) in the *Modal Title* slot on your style groups if the slot is null.

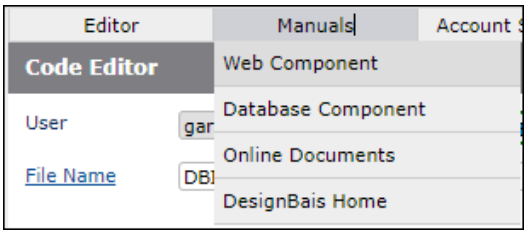


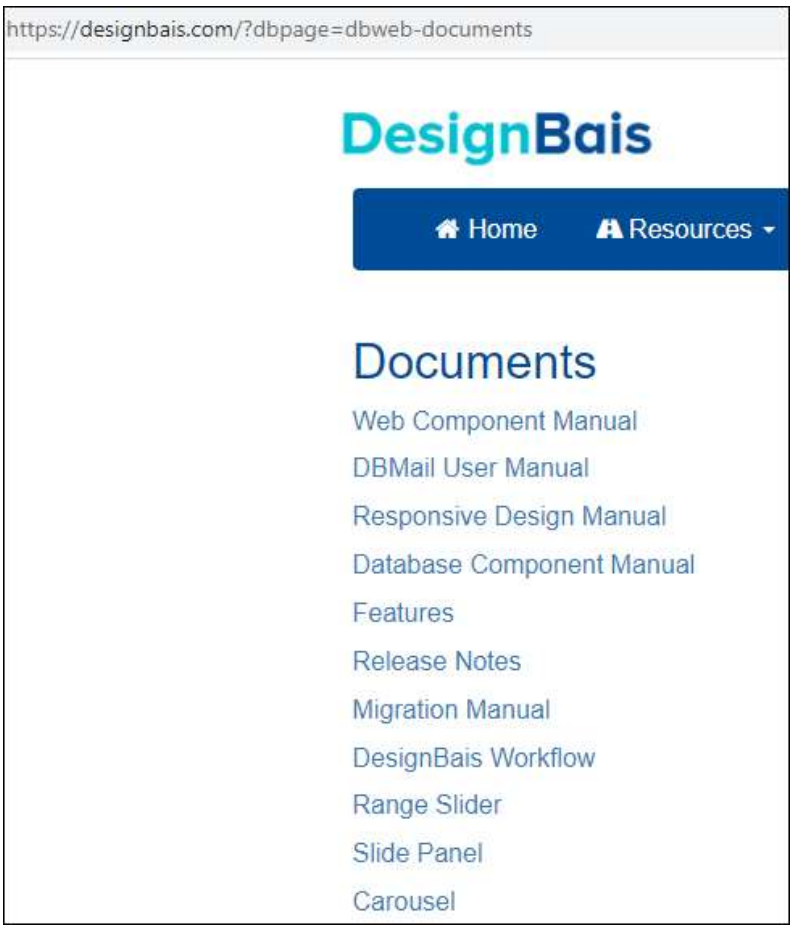
Option 19 checks DesignBais search forms and sets the button B.SEARCHDESIGNBAIS as the button action to occur when enter pressed provided there is no other button name already specified. In addition, with reference to the image below showing the text field overlapping the *Previous* button, this option locates the DBSV.PAGEDESCRIPTION text field and amends the colspan from 160 to 120. This corrects a problem whereby the left hand end of the Previous button was not sensitive to the mouse click.



Upgrading an Existing DesignBais Site

The Web Component, Database Component, Migration Manual and Release Notes Manuals are on the website under "Resources": https://www.designbais.com/?dbpage=dbweb-documents





For the Web Upgrade be careful not to overwrite files that you may have changed:

- The web.config file
- The db.config file
- The pdfcosole & dbmail configuration files
- Any standard images that you may have replaced
- Any standard css files that you may have modified
- Custom.js, admin/htmlMetaTags.txt

Suggested steps to follow:

- Be sure no one is logged in
- Stop the website or DesignBais application pool to prevent users from accessing the system during the upgrade process.
- Backup the existing website folder (DBNET)
- Download the new web component
- As a guide the following files can generally be removed from the new web component so that the existing site-modified versions will not be overwritten:

1. custom.js
2. designBaisStyle.css
3. designBaisPrint.css
4. designBaisPDF.css
5. web.config
6. admin/htmlMetaTags.txt
7. admin/db.config
8. admin/dbmail/sbmail.config
9. admin/pdfconsole/dbpdf.config
10. admin/DBSiteMonitor/config.xml

- It should not be necessary to remove any items from the rdv2 or res folders. The "sites" folders in the existing website will contain any local development files.
- If the dbmail or pdf console have changed than remember to stop the existing processes before copying over the top.
- Copy the remaining downloaded new web component folder over the existing website folder
- Restart the website

For the Database Component:

- Check that the website is not in use
- Backup accounts to be upgraded
- Download the DBINET account (or perhaps all DesignBais Release accounts in the full download)
- Copy the downloaded DBINET over the existing DBINET and load the new DBINET.DEMO release if needed
- Log to each data account and RUN DBINET DBI.P.ACCOUNT.SETUPNET to catalog any new routines & create any new files.
- In Style Definition there is a link button “Set Style Heights” which calculates FONT size information by analysing styles and using the calculated padding, borders and font height in order to create a more accurate element both in Forms Designer and at run time. This should be run in all browsers as they render fonts at different sizes. The largest calculated size is retained.
- Call up a style using the Style Definitions form on the DesingBais tools menu. Do a dummy save of the style to rebuild the css which will pick up any new/changed DesignBais styles (needed if the data component was upgraded).
- **Run upgrade routines** – see the Migration Manual for details. If moving from a release prior to 8.3.3.4 you must run Upgrade Routine 30 to adjust the heights of input boxes, check boxes & radio buttons used in your forms.
- Regenerate the CSS files from a data account after checking the local DBISTYLE has the latest “dbais” styles.

Extract SYS File Records

The elements that make up the DesignBais tool set (files, properties, forms, selects, reports and menus) are distributed in what are called *SYS* files. There are 4 files:

DBISYSFILES
DBISYSFORMS
DBISYSPROP
DBISYSSELECT

The DBISYSFORMS file is used for forms and for a range of other items:

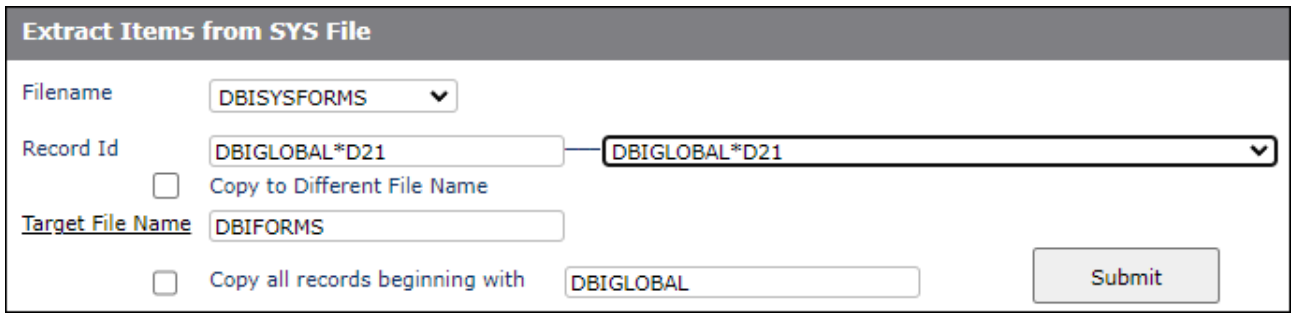
- Standard Fonts
- Form Help Records
- Menus
- Reports
- Styles
- Style Groups
- Dialog Box Defaults

If a developer wants to customize a DesignBais tool form, for example, then the form must be extracted from the DBISYSFORMS file and loaded into the local DBIFORMS file. The form can then be modified in the normal manner. It becomes part of the developer’s application.

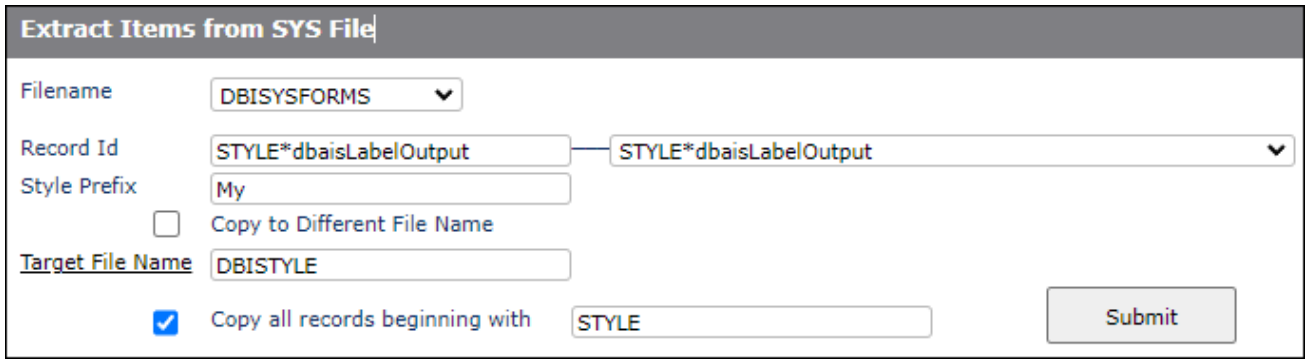
When a later release of DesignBais is loaded the modified form should be removed from the local DBIFORMS file and the local modifications re-applied to the version of the form extracted from the new DesignBais release.

The *Extract SYS File Records* option is designed to allow items from the *SYS* files to be extracted to the local files.

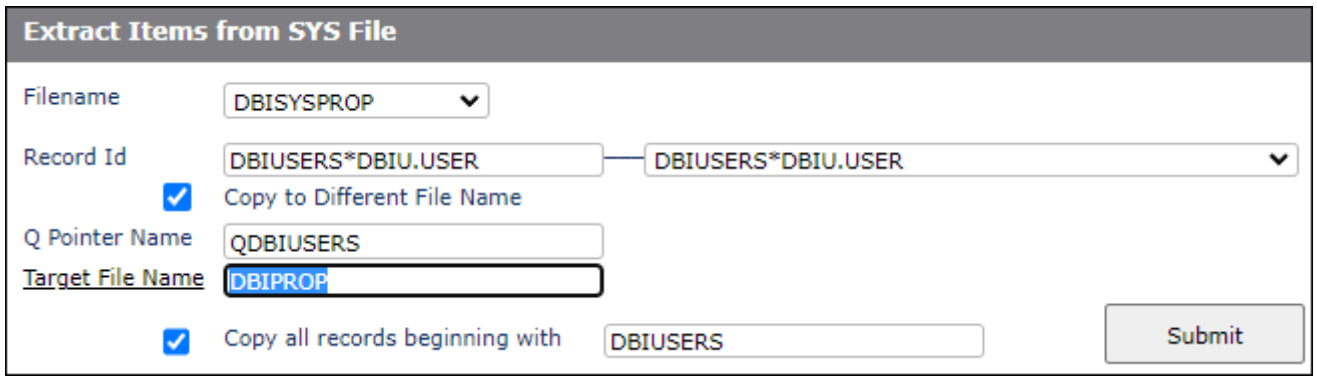
The example below shows how the Global Login form DBIGLOBAL*D21 can be extracted into the local DBIFORMS file.



In the next example the *Copy all records beginning with* check box has been ticked. This allows, in this example, all DesignBais style records to be extracted from DBISYSFORMS and loaded into DBISTYLE. DesignBais requires that the syle names be changed by replacing the *dbais* prefix with a prefix selected by the developer, in this case *My* has been selected as the new prefix.

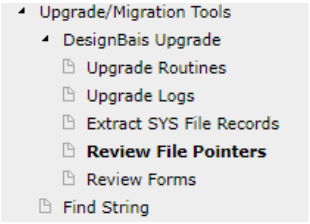


The following example shows how to change the id of the extracted items.
In this example we want to use the DesignBais DBIUSERS file in our DesignBais application. To do this we must set up a q-pointer to DBIUSERS since it is not permitted to add a DBI file to the local DBIFILES file.
We create a file called QDBIUSERS which points to DBIUSERS.
To generate the field properties we can use the following method. Click the *Copy to Different File Name* checkbox.
Enter the name of the q-pointer in the *Q Pointer Name* field, in this case the file name we have created is QDBIUSERS.

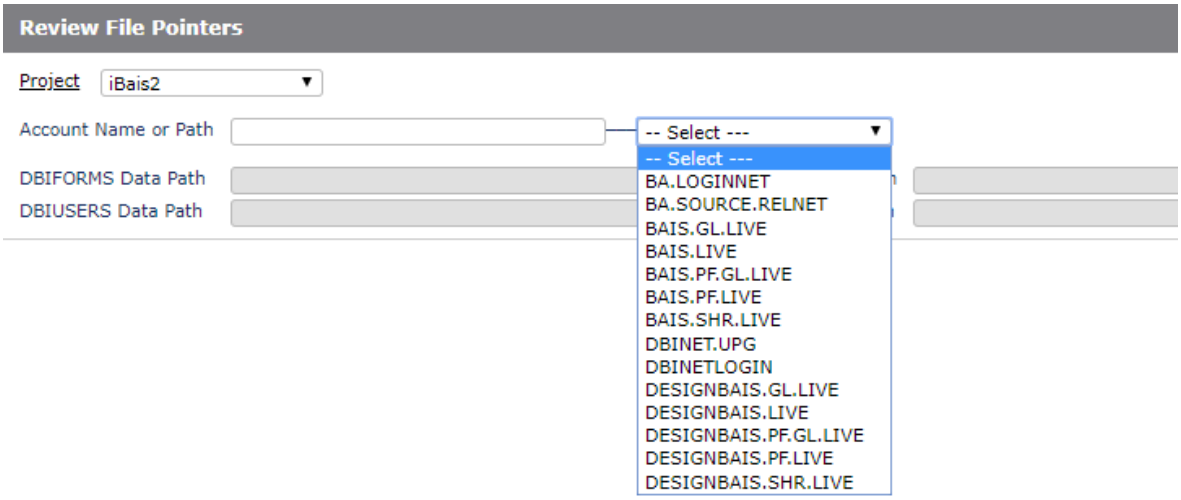


Review File Pointers

Use the “Review File Pointers” form accessible from the DesignBais Upgrade menu on the DesignBais Tools form to review file and Q pointers in a given account.

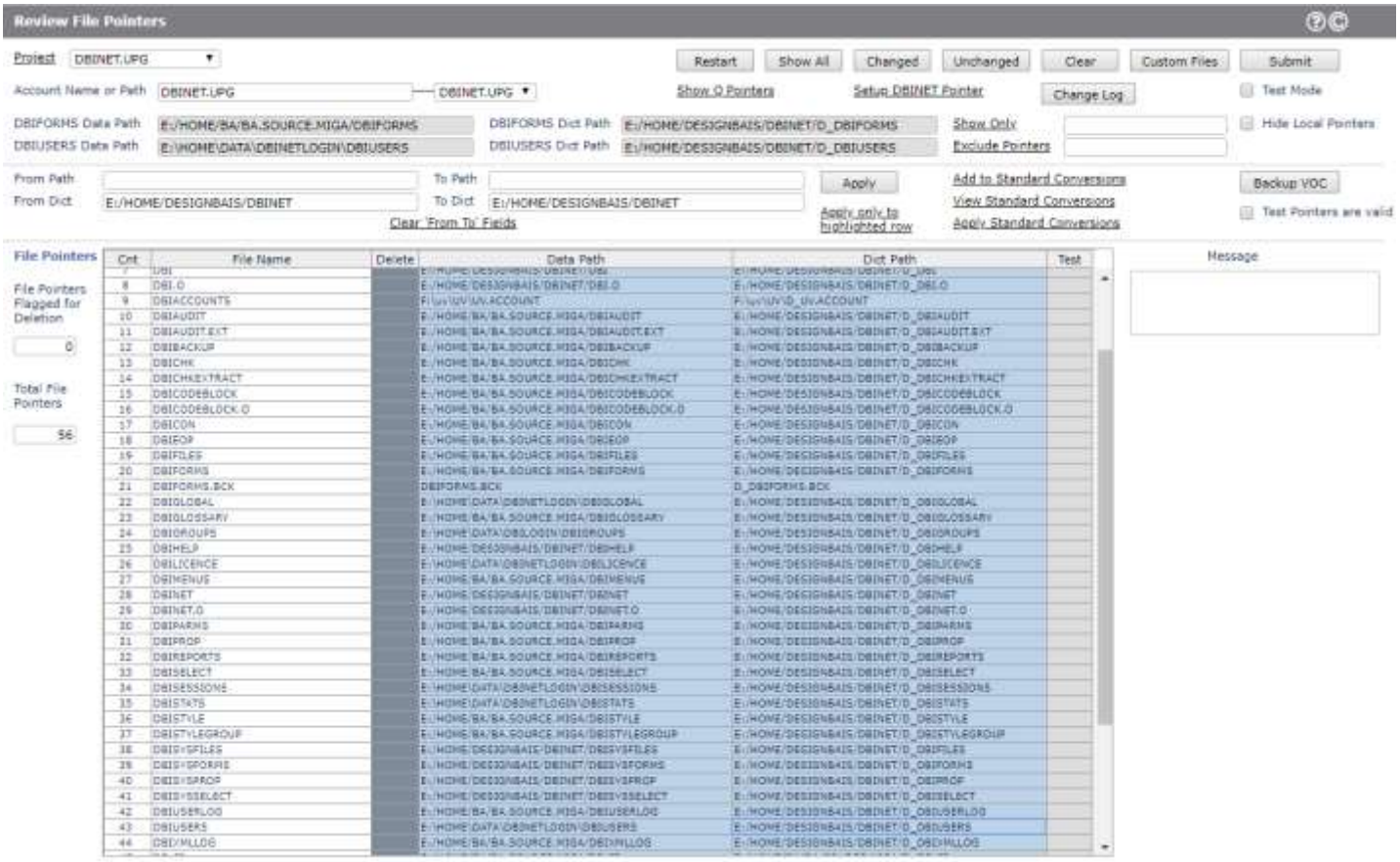


Select an existing Project from the dropdown or create a new Project by clicking the *Project* link. The selected project will determine the content of the *Account Name or Path* dropdown list.



After entering an account name or selecting an account from the list in the case that the required account has been reviewed previously, the list of Q and File pointers will be displayed.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3



File and Q pointers in the Master Dictionary of the account are listed in separate On-form Reports.

The paths to the DBIFORMS and DBIUSERS files are extracted and displayed near the top of the form. The paths to these 2 files are significant and often indicate where other files must point to.

The 'Account Name' and 'File Name' columns in the Q Pointers report can be clicked in order to load the values from the clicked row into the. Similarly the 'Data Path' and 'Dict Path' can be clicked.

It is then possible to amend the 'To Path' and 'To Dict' fields to the new location (path) that Q Pointers and File Pointers are to point to. Clicking the '*Apply'* button causes all the pointers that match the value in the 'Change Path' and 'Dict Path' to be changed to the values in the 'To Path' and 'To Dict' fields.

Use the *Apply only to highlighted row* link if the change is to effect just one row. Click the row in order to load the existing data and dict paths into the From and To fields. Make the required change to the *To Path* or *To Dict* fields. Then click the row that is to be changed. Then click the *Apply only to highlighted row* link.

The buttons at the top of the form allow you to view all pointers, pointers that have had a change applied, or pointers that are currently unchanged. Changes are confirmed only after the 'Submit' button is clicked.

Clicking the 'Add to Standard List' button moves the currently displayed values in the 'Change Path' / 'To Path' and 'Change Dict' / 'To Dict' fields to a list of Standard Conversions. These can be displayed by clicking the 'View Standard Conversions' button.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

These Standard Conversions are stored and can be invoked for many different accounts. They can be applied to an account by clicking the ‘Apply Standard Conversions’ button.

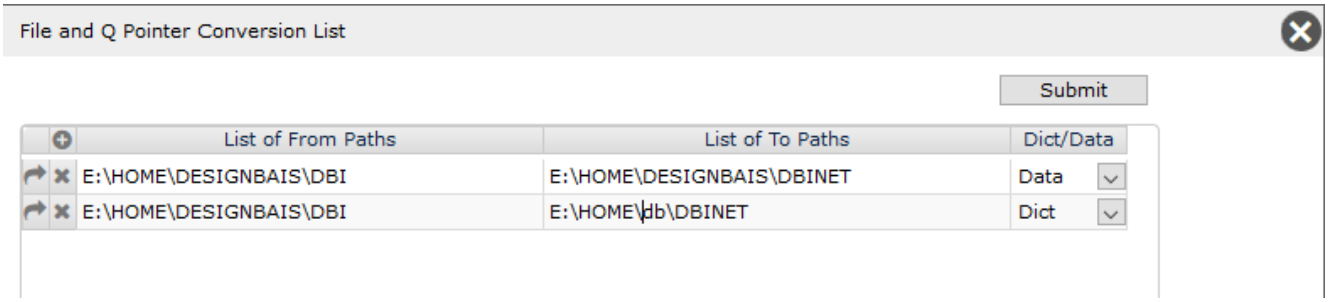
The ‘Show Only’ and ‘Exclude Pointer’ fields allow you enter a string and limit the display of pointers to only those that either match this string, or do not contain this string.

The ‘Hide Local Pointers’ check box removes local pointers from the display.

Clicking the *Delete* column allows you to select pointers to be deleted. Clicking again will toggle this action so that the pointer is retained. Deletion is actually carried out after the ‘Submit’ button is clicked. Some pointers are automatically marked for deletion. These are pointers where the name matches a particular pattern such names beginning with ‘QF.’ or ‘$EMAIL’ or ‘$MACRO’.

All these tools are designed to assist in setting pointers to a particular target account and, by filtering the display of pointers, to be able to home in on any pointers that point to external locations and need to be amended.

The snip below demonstrates a standard conversion where all pointers to ‘E:\HOME\DESIGNBAIS\DBI’ can be changed to point to ‘E:\HOME\DESIGNBAIS\DBINET’.



The ‘Test Mode’ check box can be clicked to prevent updates being confirmed when the ‘Submit’ button is clicked. In test mode the anticipated updates are displayed in a report without actually being carried out.

Form Compare

The Form Compare function allows the developer to compare two forms. Differences will be highlighted and displayed. It is primarily targeted at comparing forms that are very similar in order to reveal minor differences, or to reveal what changes have been made to a form since the last backup copy was made.

After entering the Form Filename and Form Name of each form to be compared click the “Compare” button, or just press the “Enter” key.

All DBIFORMS field properties are displayed. Clicking the “Attribute” column allows you to view the actual form record attributes.

Clicking the “Ignore” column allows you to flag form attributes that you want to ignore whan comparing forms. This is useful so that fields like “Last Updated Date” do not show up as differences between two forms. The list of “Ignore” attributes is stored, by user id, on DBIPARMS. Clicking the highlighted “Ignore” flag allows you to turned off the ignore status.

In the following example the Form Compare function is comparing form DBCLIENT*PDORIG with DBCLIENT*PDCOPY.

It shows that:

Form 2 (DBCLIENT*PDCOPY) no longer has a header form

The Last Updated By field is different

Form 1 has a read group of 1 using DBRECORD while Form 2 has read group of 2 using DBOTHER.RECORD(2)

The section name on the DBC.SUBURB field has been changed from “Main” to “Town”

There are differences in the section control details (Section Differences Between Form 1 and Form 2)

Note that for section differences you can click the “Show Details” button which displays “Section Differences” and the list of fields in each of the sections listed in the “Section Differences” report.

See example screen shots below.

DesignBais Form Compare

Form 1 Details

Account Name or Path	DB.NET
Forms on File	DBIFORMS
Form Filename	DBCLIENT
Form Name	PDORIG

Form 2 Details

Account Name or Path	DB.NET
Forms on File	DBIFORMS
Form Filename	DBCLIENT
Form Name	PDCOPY

Compare

Differences Between Form 1 and Form 2

Cnt	Attribute	Form Property Field	Form Property	Form 1 Value	Form 2 Value
1	134,2	DBIF.FORM.INCLUDE.FOOTER	Include Include Form	DBCLIENT_DEMOFOOTER	(null value)

Field Differences Between Form 1 and Form 2

Cnt	Field Name	Form Property Field	Field Description	Form Property	Form 1 Value	Form 2 Value
1	DBC.CLIENT.CODE	DBIF.FIELD.READ.VARIABLE	Client Code	Read Variable	DBRECORD	DBOTHER.RECORD(2)
2	DBC.CLIENT.CODE	DBIF.FIELD.READGROUP.LIST	Client Code	Readgroup	1	2
3	DBC.SUBURB	DBIF.FIELD.SECTION	Town/City	Field Section	Main	Town

Section Differences Between Form 1 and Form 2

Show Details

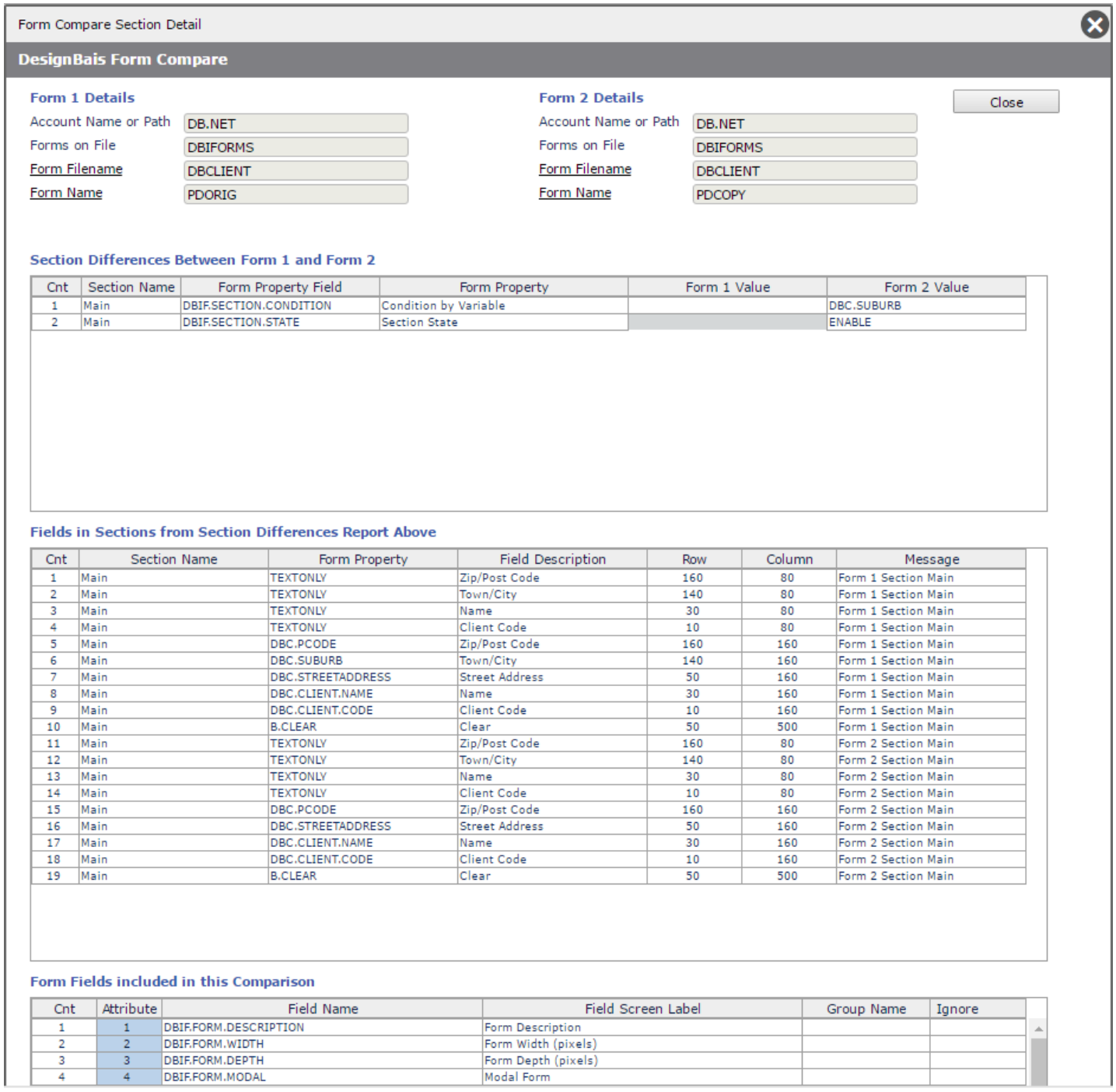
Cnt	Section Name	Form Property Field	Form Property	Form 1 Value	Form 2 Value
1	Main	DBIF.SECTION.CONDITION	Condition by Variable		DBC.SUBURB
2	Main	DBIF.SECTION.STATE	Section State		ENABLE

Sections Missing from Form 1 and Form 2

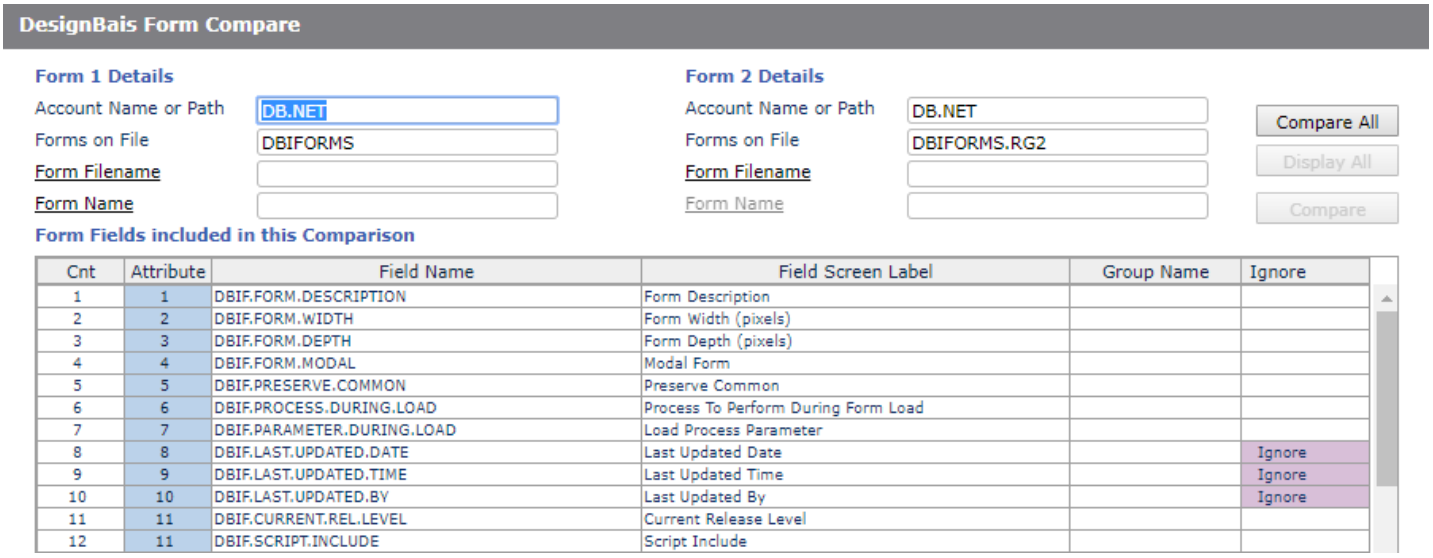
Cnt	Section Name	Form Property	Field Description	Row	Column	Message
1	Town	DBC.SUBURB	Town/City	140	160	Form 2 not on Form 1

Reads Missing Between Form 1 and Form 2

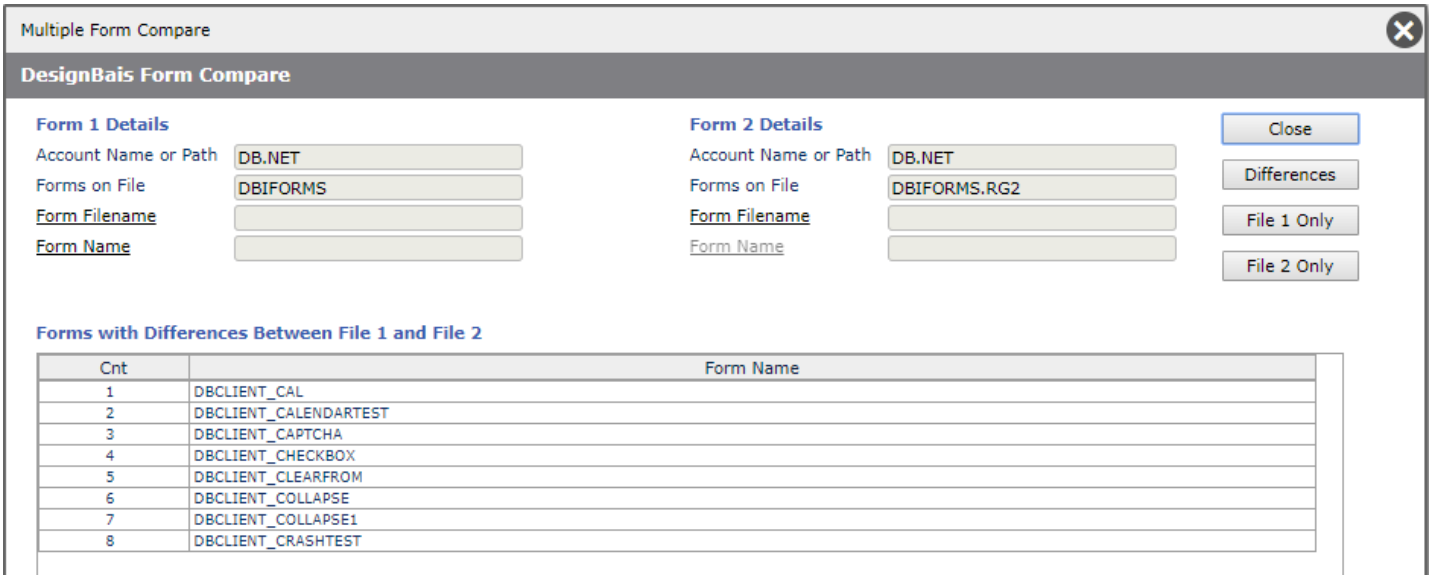
Cnt	Read Group	Field Name	Field Description	Row	Column	Message
1	1 [DBRECORD]	DBC.CLIENT.CODE	Client Code	10	160	Form 1 not on Form 2
2	1 [DBOTHER.RECORD(2)]	DBC.CLIENT.CODE	Client Code	10	160	Form 2 not on Form 1


Form Compare Section Detail
DesignBais Form Compare
Form 1 Details
Account Name or Path DB.NET
Forms on File DBIFORMS
Form Filename DBCLIENT
Form Name PDORIG
Form 2 Details
Account Name or Path DB.NET
Forms on File DBIFORMS
Form Filename DBCLIENT
Form Name PDCOPY
Close
Section Differences Between Form 1 and Form 2
Cnt | Section Name | Form Property Field | Form Property | Form 1 Value | Form 2 Value
1 | Main | DBIF.SECTION.CONDITION | Condition by Variable | | DBC.SUBURB
2 | Main | DBIF.SECTION.STATE | Section State | | ENABLE
Fields in Sections from Section Differences Report Above
Cnt | Section Name | Form Property | Field Description | Row | Column | Message
1 | Main | TEXTONLY | Zip/Post Code | 160 | 80 | Form 1 Section Main
2 | Main | TEXTONLY | Town/City | 140 | 80 | Form 1 Section Main
3 | Main | TEXTONLY | Name | 30 | 80 | Form 1 Section Main
4 | Main | TEXTONLY | Client Code | 10 | 80 | Form 1 Section Main
5 | Main | DBC.PCODE | Zip/Post Code | 160 | 160 | Form 1 Section Main
6 | Main | DBC.SUBURB | Town/City | 140 | 160 | Form 1 Section Main
7 | Main | DBC.STREETADDRESS | Street Address | 50 | 160 | Form 1 Section Main
8 | Main | DBC.CLIENT.NAME | Name | 30 | 160 | Form 1 Section Main
9 | Main | DBC.CLIENT.CODE | Client Code | 10 | 160 | Form 1 Section Main
10 | Main | B.CLEAR | Clear | 50 | 500 | Form 1 Section Main
11 | Main | TEXTONLY | Zip/Post Code | 160 | 80 | Form 2 Section Main
12 | Main | TEXTONLY | Town/City | 140 | 80 | Form 2 Section Main
13 | Main | TEXTONLY | Name | 30 | 80 | Form 2 Section Main
14 | Main | TEXTONLY | Client Code | 10 | 80 | Form 2 Section Main
15 | Main | DBC.PCODE | Zip/Post Code | 160 | 160 | Form 2 Section Main
16 | Main | DBC.STREETADDRESS | Street Address | 50 | 160 | Form 2 Section Main
17 | Main | DBC.CLIENT.NAME | Name | 30 | 160 | Form 2 Section Main
18 | Main | DBC.CLIENT.CODE | Client Code | 10 | 160 | Form 2 Section Main
19 | Main | B.CLEAR | Clear | 50 | 500 | Form 2 Section Main
Form Fields included in this Comparison
Cnt | Attribute | Field Name | Field Screen Label | Group Name | Ignore
1 | 1 | DBIF.FORM.DESCRIPTION | Form Description
2 | 2 | DBIF.FORM.WIDTH | Form Width (pixels)
3 | 3 | DBIF.FORM.DEPTH | Form Depth (pixels)
4 | 4 | DBIF.FORM.MODAL | Modal Form

There is also an option to compare all forms in 2 different form files. In the example below a different file name is entered in the Form 2 Details. This enables the “Compare All” button. Clicking this button will then run a phantom process that compares all forms in the 2 files and displays the results in 3 reports.

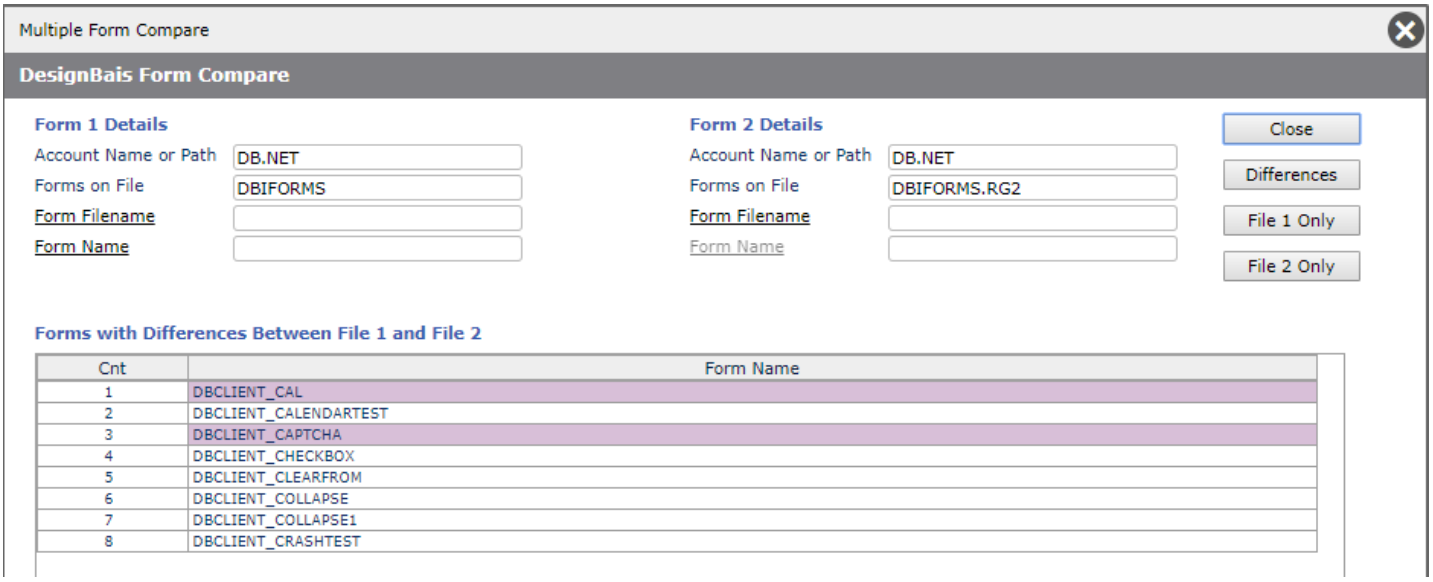


Those forms that exist in both files and are different are displayed in the Differences report.

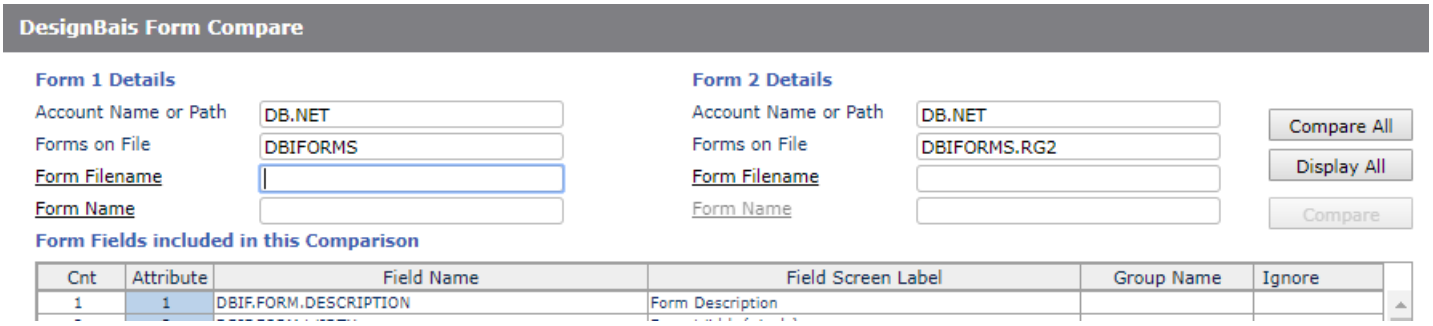


Forms that exist only in File 1 or File 2 are displayed in reports that can be displayed by clicking the “File 1 Only” or “File 2 Only” buttons.

Clicking the “Differences” button displays or re-displays the list of forms that are different. Clicking the Form Name column displays the details of the differences. Click the close button to return to the list of forms with differences. As each form is reviewed the report column color changes.



After closing the Multiple Form Compare form the “Display All” button is enabled. This will re-display the report of Differences without re-running the phantom process.

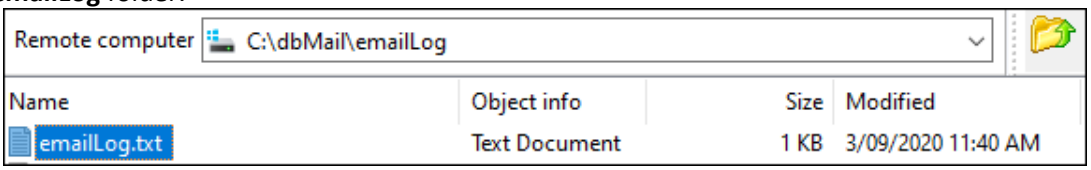


DBMail

DBMail is a Windows .NET program that generates email messages from **xeml** files dropped in an input folder and sends them out to specified recipients.

DBMail checks its input folder at regular intervals (5 seconds [internal]) to see if there are any xeml files to be processed.

DBMail has no user interface. It's started by double clicking and stopped by dropping a file named "stop.txt" in the folder that it runs in. In most cases you'll want to run it as a Windows scheduled task. All activity is logged in the **emailLog** folder.

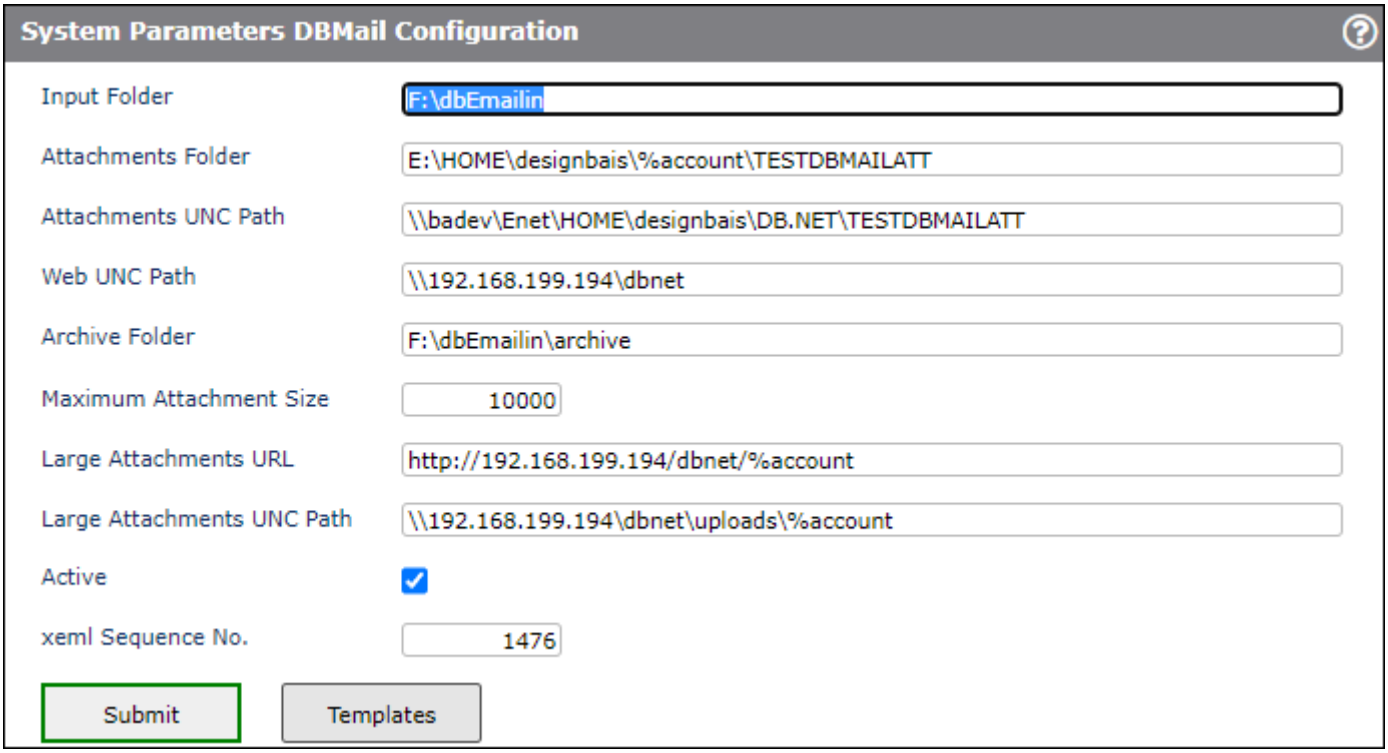


DBMail supports:

- html formatted messages accompanied with images and css files
- attachment compression (zip format)
- CSV to Excel conversion of attachments
- HTML to PDF conversion of attachments

Refer to the full User Manual available on the website.

On the database there is a configuration form accessed from the System Parameters menu. The parameters are held on DBIPARMS with a record id *DBMAIL*.



Input Folder	The Input Folder will contain the ".xeml" files that define the email contents for the dbMail server. This must be a database table name that points to an operating system folder that is accessible by the dbMail server.
Attachments Folder	This is a database table name that points to an operating system folder that is accessible from the dbMail server. Attachments for emails must be placed in this folder. Reports generated by DesignBais will be written to this folder for emailing by the dbMail server. Attachments are moved from folder to an archive folder after the email is sent. Do not use attachment paths that are significant elsewhere in your deployment. Attachment paths pointing to a web site's files may not be a good idea.
Attachments UNC Path	This is the UNC file path that points to the same operating system folder that is to be used for attachments for emails generated in this database account. This path is how the dbMail server will access attachments for emails.
Web UNC Path	This is the UNC file path that points to the DesignBais website root directory in order to access the DesignBaisPrint.css and image files included on reports. This path is how the dbMail server will access these files for emails. If the website is not used directly then the css & image files will need to be copied to the chosen path.
Achive Folder	Email archives are placed in this folder.
Maximum Attachment Size	The maximum size in kilobytes for attachments. Larger files will not be emailed but will be made available via a download link.
Large Attachments URL	This is the URL of the attachment file. When the file exceeds the *Maximum Attachment Size* it is not sent as an attachment but, rather, this link is added to the email as a hyperlink.
Large Attachments UNC Path	This UNC path determines the target folder to which the large file is transferred to be served as a link to the user. Please note that unauthorised access to this folder needs to be restricted for security reasons – e.g., uploads folder. Authorised access can be applied to any folder with a change to web.config to enable the HTTP handler.
Active	If dbMail is active then report and other emails generated by DesignBais will be sent using a dbMail server.
xeml Sequence No	Sequential number for the xeml file name when no filename is provided by the application.

When any of the parameters *Maximum Attachment Size, Large Attachments URL* or *Large Attachments UNC Path* are missing, the files will be sent as attachments regardless of size.

The subroutine DBI.G.DBMAIL can be called to format the .xeml file for input to DBMail. This subroutine has one argument which is a dynamic array as shown in the example below. The resulting .xeml file is written to the *Input Folder* specified in the *DBMail Configuration* form.

Important Note:

- EMAIL.FROM must contain an email address.
- EMAIL.FROM.NAME may contain text such as the name of the sender.

```
DBI.G.DBMAIL(DBMAIL.REC)
      XEML.ID           = DBMAIL.REC<DBM.XEML.ID>
      LOG.LABEL         = DBMAIL.REC<DBM.LOG.LABEL>
      EMAIL.SUBJECT     = DBMAIL.REC<DBM.SUBJECT>
      EMAIL.MESSAGE     = DBMAIL.REC<DBM.MESSAGE>
      EMAIL.FORMAT      = DBMAIL.REC<DBM.FORMAT>
      EMAIL.TEMPLATE    = DBMAIL.REC<DBM.TEMPLATE>
      EMAIL.FIELD       = DBMAIL.REC<DBM.TEMP.FIELD>
      EMAIL.VALUE       = DBMAIL.REC<DBM.TEMP.VALUE>
      EMAIL.TO          = DBMAIL.REC<DBM.TO>
      EMAIL.FROM        = DBMAIL.REC<DBM.FROM>
      EMAIL.FROM.NAME   = DBMAIL.REC<DBM.FROM.NAME>
      EMAIL.REPLYTO     = DBMAIL.REC<DBM.REPLY.TO>
      EMAIL.SENDER      = DBMAIL.REC<DBM.SENDER>
      EMAIL.CC          = DBMAIL.REC<DBM.CC>
      EMAIL.BCC         = DBMAIL.REC<DBM.BCC>
      EMAIL.ATT         = DBMAIL.REC<DBM.ATTACHMENT>
      EMAIL.CONV        = DBMAIL.REC<DBM.ATT.CONVERT>
      EMAIL.ZIP         = DBMAIL.REC<DBM.ATT.ZIP>
      PDF.LAYOUT        = DBMAIL.REC<DBM.PDF.LAYOUT>
      XLS.DELIM         = DBMAIL.REC<DBM.XLS.DELIMITER>
      XLS.WKS           = DBMAIL.REC<DBM.XLS.WKS>
      RES.ID            = DBMAIL.REC<DBM.RESOURCE.ID>
      RES               = DBMAIL.REC<DBM.RESOURCE>
      RES.SRC           = DBMAIL.REC<DBM.RESOURCE.SRC>
      FIRST.HDR         = DBMAIL.REC<DBM.FIRST.HDR>
      XLS.TABLE         = DBMAIL.REC<DBM.XLS.TABLE>
      XLS.CULTURE       = DBMAIL.REC<DBM.XLS.CULTURE>
      DATE.FORMAT       = DBMAIL.REC<DBM.DATE.FORMAT>
      ROW.DELIM         = DBMAIL.REC<DBM.ROW.DELIMITER>
      XLS.TEXT.DELIM    = DBMAIL.REC<DBM.XLS.TXT.DELIM>
      XLS.TXT.ENC       = DBMAIL.REC<DBM.XLS.TXT.ENC>
      COLUMN.TYPE       = DBMAIL.REC<DBM.XLS.COLUMN.TYPE>
      TIME.FORMAT       = DBMAIL.REC<DBM.XLS.TIME.FORMAT>
      DECIMALS          = DBMAIL.REC<DBM.XLS.DECIMALS>
```

The equates for DBI.G.DBMAIL are:

```
* Equates for the DBI.G.DBMAIL Argument
*
* Hand written - DO NOT ALTER
*
EQU DBM.XEML.ID         TO  1   ;* xeml file name - if blank a sequential number will be used
EQU DBM.LOG.LABEL       TO  2   ;* string that appears in logs and identifies a message
EQU DBM.SUBJECT         TO  3   ;* email subject
EQU DBM.MESSAGE         TO  4   ;* email body as HTML or TEXT
EQU DBM.FORMAT          TO  5   ;* HTML or TEXT default is HTML
EQU DBM.TEMPLATE        TO  6   ;* DBMAIL*TemplateName as configured on DBIPARMS
EQU DBM.TEMP.FIELD      TO  7   ;* Field name appearing in the template e.g. $$NAME$$ to be replaced by the value
EQU DBM.TEMP.VALUE      TO  8   ;* Field value
EQU DBM.TO              TO  9   ;* Can be multivalued or delimited by semi-colons email addresses
EQU DBM.FROM            TO 10   ;* email address of sender
EQU DBM.FROM.NAME       TO 11   ;* name of sender
EQU DBM.REPLY.TO        TO 12   ;* email address for replies
EQU DBM.SENDER          TO 13   ;* email address of domain
EQU DBM.CC              TO 14   ;* email address for copy may be delimited by semi-colons
EQU DBM.BCC             TO 15   ;* email address for blind copy may be delimited by semi-colons
EQU DBM.ATTACHMENT      TO 16   ;* multivalued list of attachment files (path is from System Parameters)
EQU DBM.ATT.CONVERT     TO 17   ;* PDF or XLS or blank
EQU DBM.ATT.ZIP         TO 18   ;* true or false if the attachment is to be zipped before sending
EQU DBM.PDF.LAYOUT      TO 19   ;* layout for PDF = P for portrait or L for Landscape
EQU DBM.XLS.DELIMITER   TO 20   ;* field delimiter in source file for XLS conversion
EQU DBM.XLS.WKS         TO 21   ;* worksheet name
EQU DBM.RESOURCE.ID     TO 22   ;* multivalued cid names from HTML message
EQU DBM.RESOURCE        TO 23   ;* multivalued cid file names for HTML message
```

```
EQU DBM.RESOURCE.SRC    TO 24   ;* multivalued cid location A=Attachments Folder W=Web Site folder
EQU DBM.FIRST.HDR       TO 25   ;* First Row is Header flag "true", "Y", "1" or "false", "N", "0".
                                   No value will be "true"
EQU DBM.XLS.TABLE       TO 26   ;* Excel Table Format "N" or "None" for raw data. Valid style for Excel Table (see
                                   dropdown). No value will use parameters.
EQU DBM.XLS.CULTURE     TO 27   ;* Culture for Date Identification. No value will use parameters.
EQU DBM.DATE.FORMAT     TO 28   ;* Caret delimited per column. dd-MM-yyyy or MM-dd-yyyy. If the value in a column is null
                                   then the 0 or 1 from System or Global parameters is used for "Date Format".
EQU DBM.ROW.DELIMITER   TO 29   ;* Row delimiter for ASCI file. Default is 10, may use 13.
EQU DBM.TEMP.REPL.SEQ   TO 30   ;* The replacement sequence flag associated with TEMP.FIELD and TEMP.VALUE
EQU DBM.XLS.TXT.DELIM   TO 31   ;* Text surround character e.g. 34 for '"'
EQU DBM.XLS.TXT.ENC     TO 32   ;* Text encoding- DEFAULT for ASCII or UTF8 for utf-8. Introduced for
                                   Currency Conversions.
EQU DBM.XLS.COLUMN.TYPE TO 33   ;* Caret delimited string S=string, D=date, T=time, N=numeric
EQU DBM.XLS.TIME.FORMAT TO 34   ;* Caret delimited time formats e.g. 24hr = HH:mm:ss, 12hr = hh:mm:sstt - ss is optional
EQU DBM.XLS.DECIMALS    TO 35   ;* Caret delimited per column shouwing the number of decimals for numeric fields
```

Note that attributes 28, 33, 34 and 35 may also be VM delimited. The VM will be converted to caret (^) to pass to the web.

There are several entry points that can be used in calling DBI.G.DBMAIL as illustrated by the code snippet below.

```
READ DBM FROM F.DBIPARMS,"DBMAIL" ELSE DBM = ''
IF DBM<DBIPM.DBM.ACTIVE> # "Y" AND DBIGLOBAL THEN
   READ DBM FROM F.DBIGLOBAL,"DBMAIL" THEN
      USE.GLOBAL = 1
   END ELSE
      DBM = ''
   END
END
DBM = CHANGE(DBM,"%account",DBIACCOUNT)
* Set entry point flags
ACTIVE.CHECK = 0
OPEN.ATT = 0
BEGIN CASE
   CASE DBMAIL.REC<DBM.XEML.ID> = 'ACTIVE.CHECK'
      * Use this code to verify that DBMail is Active - IERR.TEXT will be set
      ACTIVE.CHECK = 1
   CASE DBMAIL.REC<DBM.XEML.ID> = 'OPEN.ATT'
      * Use this code to open the attachments file to write out a file record
      OPEN.ATT = 1
   CASE DBMAIL.REC<DBM.XEML.ID> = 'CHECK.ATT'
      ATT.FILE = CHANGE(DBVALUE,"%account",DBIACCOUNT)
      GOSUB OPEN.ATT
      RETURN
   CASE DBMAIL.REC<DBM.XEML.ID> = 'CHECK.INPUT'
      INPUT.FILE = CHANGE(DBVALUE,"%account",DBIACCOUNT)
      GOSUB OPEN.INPUT
      RETURN
   CASE DBMAIL.REC<DBM.XEML.ID> = 'CHECK.ARCHIVE'
      ARCHIVE.FILE = CHANGE(DBVALUE,"%account",DBIACCOUNT)
      GOSUB OPEN.ARCHIVE
      RETURN
   CASE DBMAIL.REC<DBM.XEML.ID> = 'OPEN.ARCHIVE'
      ARCHIVE.FILE = CHANGE(DBM<DBIPM.DBM.ATT>,"%account",DBIACCOUNT)
      GOSUB OPEN.ARCHIVE
      IF IERR.TEXT = "" THEN DBMAIL.REC = F.ARCHIVE.FILE
      RETURN
   CASE DBMAIL.REC<DBM.XEML.ID> = 'WEB.PATH'
      WEB.PATH = DBM<DBIPM.DBM.WEB.PATH>
      IF WEB.PATH#'' AND WEB.PATH[LEN(WEB.PATH),1] # "\" THEN WEB.PATH := "\"
      DBMAIL.REC = WEB.PATH
      RETURN
   CASE DBMAIL.REC<DBM.XEML.ID> = 'ATT.PATH'
      ATT.PATH = DBM<DBIPM.DBM.ATT>
      IF ATT.PATH#'' AND ATT.PATH[LEN(ATT.PATH),1] # "\" THEN ATT.PATH := "\"
      DBMAIL.REC = ATT.PATH
      ATT.PATH = DBM<DBIPM.DBM.ATT.PATH>
      IF ATT.PATH#'' AND ATT.PATH[LEN(ATT.PATH),1] # "\" THEN ATT.PATH := "\"
      DBMAIL.REC<2> = ATT.PATH
      RETURN
```

```
END CASE
```

Pass the parameter via DBMAIL.REC<DBM.XEML.ID>
If the open request is successful the file handle of the opened file will be returned in DBMAIL.REC.
If the action is not successful then IERR.TEXT will contain the error as shown below.

Unless noted otherwise the following calls to DBI.G.DBMAIL obtain the folder to be opened from the read of the parameter record from DBIPARMS record id *DBMAIL*.
Note that the literal "%account" in any of the DBMail paths will be replaced by the value in DBIACCOUNT.

Parameter	Result
ACTIVE.CHECK	If IERR.TEXT is null then DBMail is active
CHECK.ATT	The attachments folder is passed in via DBVALUE Returns the open file handle to the attachments folder in DBMAIL.REC<1> Open fail: IERR.TEXT = "[dbMail] Attachments Folder [":ATT.FILE:"] is not available"
CHECK.INPUT	The input folder is passed in via DBVALUE Returns the open file handle to the input folder in DBMAIL.REC<1> Open fail: IERR.TEXT = "[dbMail] Input Folder [":INPUT.FILE:"] is not available"
CHECK.ARCHIVE	Returns the open file handle to the archive folder in DBMAIL.REC<1> Open fail: IERR.TEXT = "[dbMail] archive Folder [":ARCHIVE.FILE:"] is not available"
OPEN.ATT	Returns the open file handle to the attachments folder in DBMAIL.REC<1> Open fail: IERR.TEXT = "[dbMail] Attachments Folder [":ATT.FILE:"] is not available"
OPEN.ARCHIVE	Returns the open file handle to the archive folder in DBMAIL.REC<1> Open fail: IERR.TEXT = "[dbMail] archive Folder [":ARCHIVE.FILE:"] is not available"
WEB.PATH	Returns the web path in DBMAIL.REC<1> Note that the web path will have a trailing backslash character (\) regardless of the setting in the DBMail parameter record.
ATT.PATH	Returns the attachments folder in DBMAIL.REC<1> Returns the attachments path in DBMAIL.REC<1>

As shown in this code snippet from DBI.G.DBMAIL developers should note the following regarding email attachments:

- The string "%web\" at the start of the attachment path will be replaced by the full UNC attachment path.
- If providing the full path to the attachment the path must begin with "\\".
- If the path does not begin with "\\" then the attachment path from the DBMail parameter record is prefixed to the path passed to DBMail.

```
ATTMNT = EMAIL.ATT<1,J>
BEGIN CASE
   CASE ATTMNT[1,5] = "%web\"    ;* replace %web\ with web UNC path which has trailing "\"
      ATTMNT = WEB.PATH:ATTMNT[6,LEN(ATTMNT)-5]
   CASE ATTMNT[1,2] # "\\"       ;* full path NOT provided
      ATTMNT=ATT.PATH:ATTMNT
END CASE
```

When using DBMAIL to send attachment files note the following:

- An attachment file name (regardless of the extension) must not match the file name of a resource. The email will not be sent and the DBMail log file will record and error similar to:

  XEML error in \\BADEV\Fnet\dbEmailin\DB.NET\DB.NET-1984.xeml. Check if all attachments names are unique without the extensions....1/08/2023 10:48:36 AM

- The attachment name must be a file type of ".csv" if using the *XLS* conversion option (REC<DBM.ATT.CONVERT> = 'XLS')
- If the field separation character in the csv file is a comma and each field is surrounded by double quotes then set REC<DBM.XLX.TXT.DELIM,n> = 34. This informs DBMail that each field is surrounded by CHAR(34) and DBMail will remove these double quotes when converting the file to xlsx format.
- Note that REC<DBM.XLX.TXT.DELIM> is multi-valued associated with DBM.ATTACHMENT.

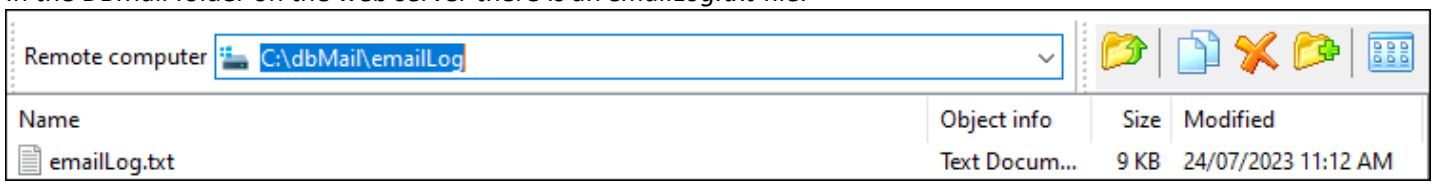
Example of the HTML generated for a file to be sent using DBMail:

```
<attachment convert="PDF" zip="false">
            <attPath><![CDATA[C:\dbmailtest\attachments\A_TEST.html]]></attPath>
            <insetX>40</insetX>
            <insetY>40</insetY>
            <magnifyX>1</magnifyX>
            <magnifyY>1</magnifyY>
            <anchorX>120</anchorX>
            <anchorY>670</anchorY>
            <rotateAngle>0</rotateAngle>
            <rotateAnchorX>400</rotateAnchorX>
            <rotateAnchorY>400</rotateAnchorY>
            <pageRotateAngle>0</pageRotateAngle>
            <maxPDFAttachmentSize>100000</maxPDFAttachmentSize>
            <largeAttachmentUrl>http://localhost/DesignBaisNet/uploads/A_TEST.PDF</largeAttachmentUrl>
            <largeAttachmentPath>\192.68.199.103\CNet\DesignBaisNet\DesignBaisNet\Uploads\A_TEST.PDF</largeAttachmentPath>
</attachment>
```

DBMail Logs

Developers can review the DBMail logs in order to track down issues with emails not appearing.

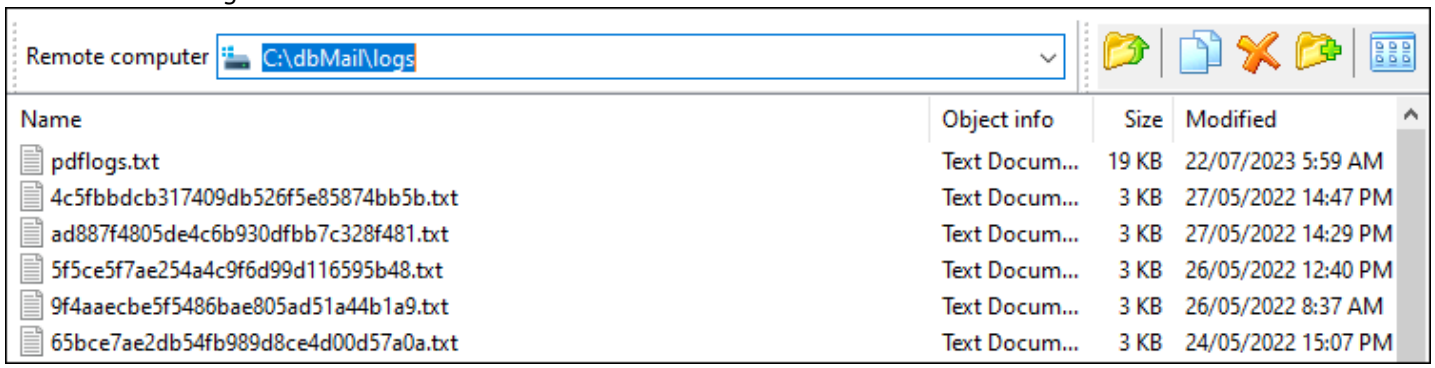
In the DBMail folder on the web server there is an *emailLog.txt* file.



Example log entries:

```
Error in moving attachments from input folder to C:\dbMail\process.  The xeml file \\BADEV\Fnet\dbEmailin\DB.NET
\DB.NET-1952.xeml may be retried later.File '\\192.168.199.194\dbnet\images\workflow\smallRedCar.jpg' not
found....22/07/2023 1:25:42 PM
Error in moving attachments from input folder to C:\dbMail\process.  The xeml file \\BADEV\Fnet\dbEmailin\DB.NET
\DB.NET-1952.xeml may be retried later.File '\\192.168.199.194\dbnet\images\workflow\smallRedCar.jpg' not
found....22/07/2023 1:37:47 PM
Error in moving attachments from input folder to C:\dbMail\process.  The xeml file \\BADEV\Fnet\dbEmailin\DB.NET
\DB.NET-1952.xeml may be retried later.File '\\192.168.199.194\dbnet\images\workflow\smallRedCar.jpg' not
found....22/07/2023 4:37:51 PM
FAILURE IN PROCESSING -  - DB.NET-1953.xeml.  Error:The specified string is not in the form required for an e-mail
address....24/07/2023 11:04:01 AM
OK -  - ...****......jon@bais.com.au......24/07/2023 11:04:01 AM
OK -  - ......DB.NET-1954.xeml...bob2@bais.com.au......24/07/2023 11:12:39 AM
```

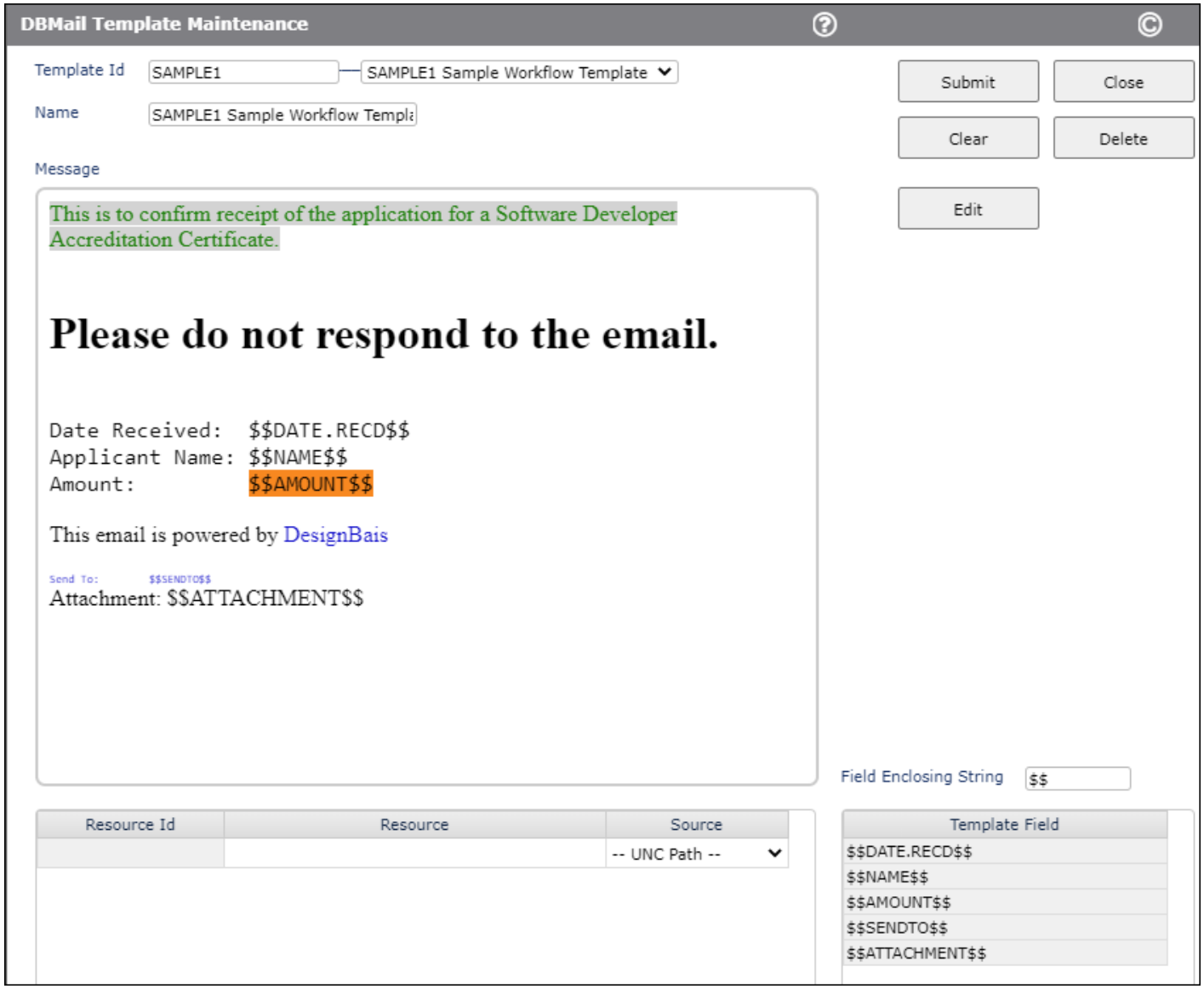
There is also a a *logs* folder.



Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

DBMail Template

A DBMail Template provides a means of creating HTML formatted emails.



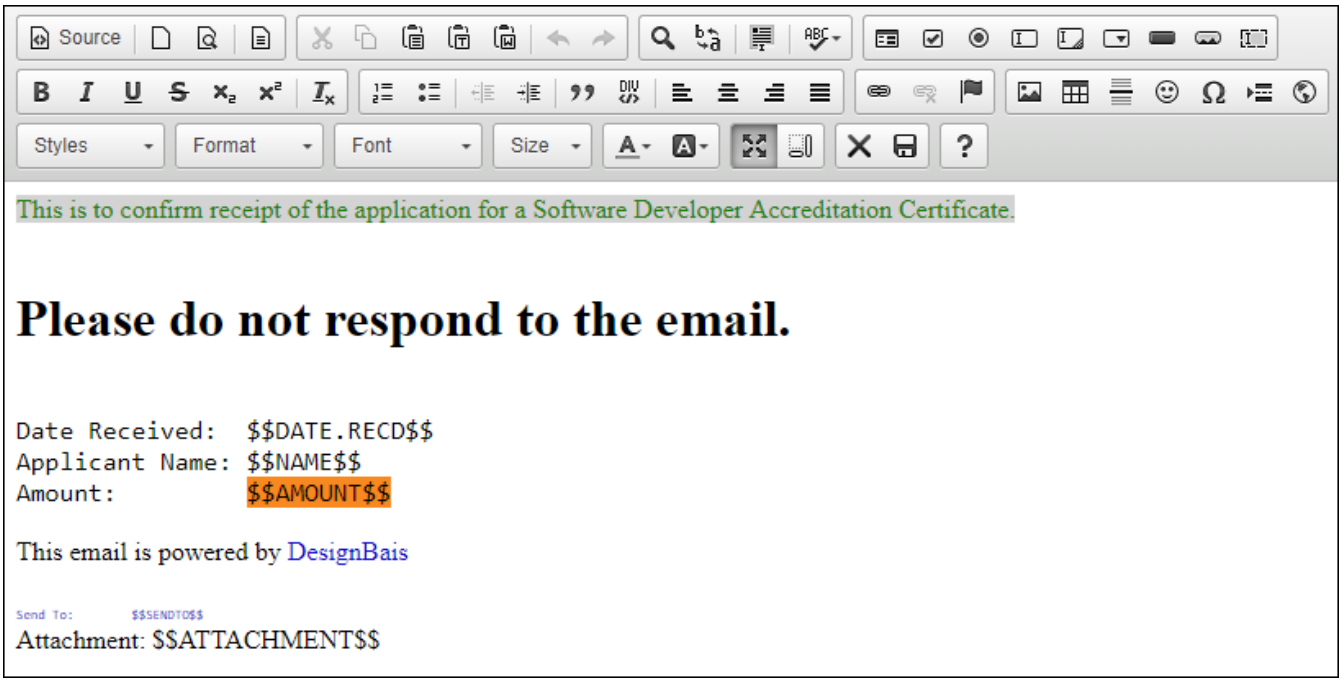
Template Id — Assign a code to identify the template.

Name — A short description of the template.

Message — The message is the body of the email. It is maintained by clicking the *Edit* button on the right of the message box. This opens the HTML editor. Use the features of this editor to create the email. Text style, font, color, tables, images can be utilized.

You can assign your own *Field Enclosing String* in order to mark fields for replacement at run time when an email is generated.

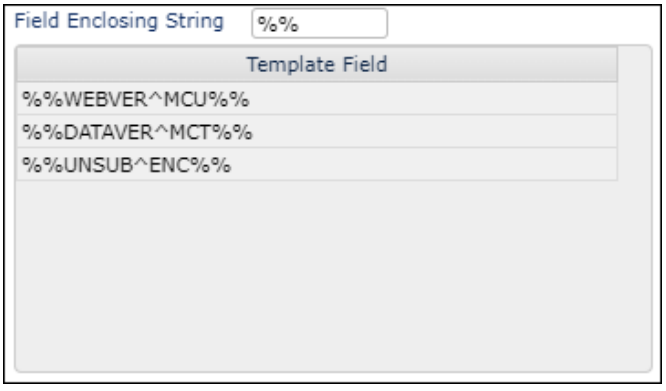
In the example below the *Field Enclosing String* is "$$". The field names enclosed by this string are extracted and listed in the *Template Field* grid box after you enter the string you have used.



So the sequence to follow is:

- Open the HTML Editor by clicking the *Edit* button.
- Enter the email message text.
- Enter replacement field names surrounded by the string you have selected, such as "$$" in our example.
- Note that this string must not exist other than as the field enclosing string.
- Save the message and exit the HTML editor.
- Enter your selected string, such as "$$", in the field called *Field Enclosing* String.
- The list of replacement field names will appear in the *Template Field* grid.

You can mask the replacement fields by using the following convention.



Append the caret character followed by any valid output conversion such as MCU, MCL or MCT to the replacement string as shown in the snip to the right.

Adding *^ENC* provides a method to scramble the data that is contained in the replacement string when creating a url.

The data to be scrambled must be preceeded by the query string pair *dbparams=link* as in the following example:

http://192.168.199.194/dbnet/?ac=ws&dbpage=dbweb-unsubscribe&dbparams=link&usr=12345

In this case the string *usr=12345* will be scrambled. The basic code to achieve this is shown following.

The first code example shows how to create the scrambled query string pair:

```
IF EMASK = 'ENC' THEN
   MARKER = 'dbparams=link&'
   LM = LEN(MARKER)
   POS1 = INDEX(USE.FIELD,MARKER,1) + LM
   * scramble whatever follows 'dbparams=link&'
   FLD2 = USE.FIELD[POS1,LEN(USE.FIELD) - POS1 + 1]
   FLDSCRAM = FIELD(FLD2,'=',2)
   CALL DBI.G.ENCODENET(FLDSCRAM,'SCRAMBLE')
   USE.FIELD = USE.FIELD[1,POS1-1]:FIELD(FLD2,'=',1):'=':FLDSCRAM
   IF TESTING THEN CRT 'USE.FIELD=':USE.FIELD
END
```

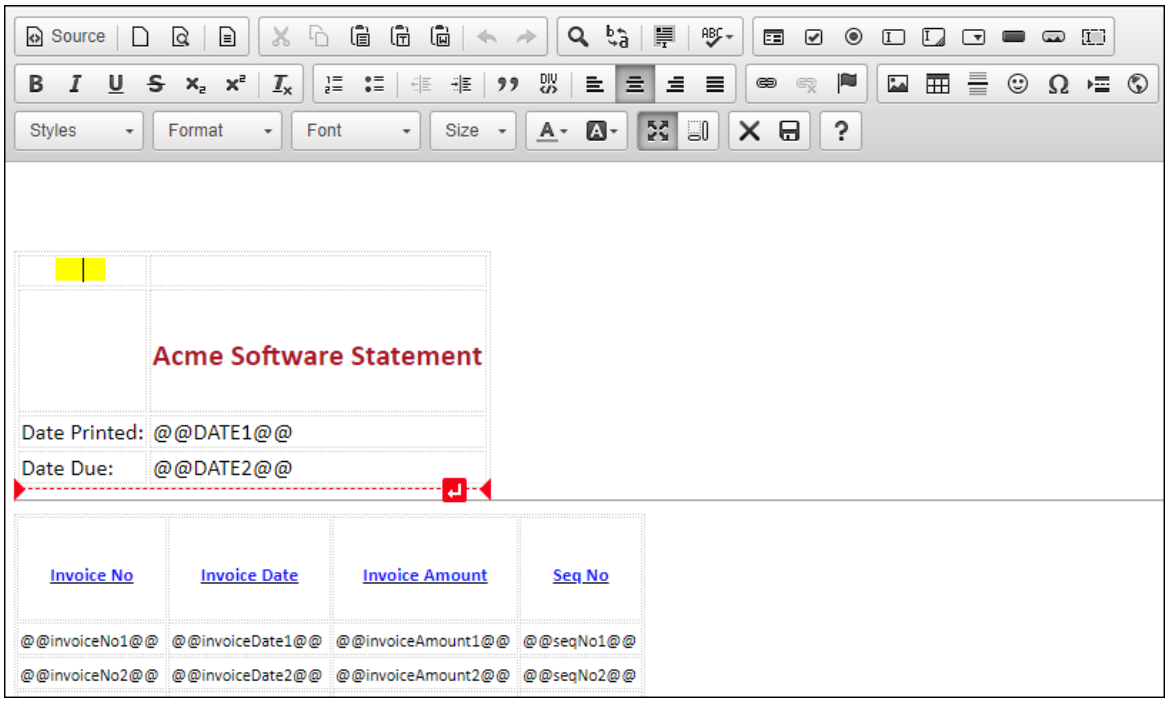
The second code example shows how to unscramble the query string pair:

```
IF DBSTORE(3)<1> = '' THEN
   MARKER = 'dbparams=link&'
   LM = LEN(MARKER)
   POS1 = INDEX(DBW3CQSTRING,MARKER,1) + LM
   FLD2 = DBW3CQSTRING[POS1,LEN(DBW3CQSTRING) - (POS1-1)]
   FLDSCRAM = FIELD(FLD2,'=',2)
   CALL DBI.G.ENCODENET(FLDSCRAM,'UNSCRAMBLE')
   FLD2 = FIELD(FLD2,'=',1):'=':FLDSCRAM
   USRFLG = FIELD(FLD2,'=',1)
   USRCDE = FIELD(FLD2,'=',2)
   IF USRFLG = 'usr' AND USRCDE # '' THEN
      DBSTORE(3)<1> = USRCDE
   END
END
```

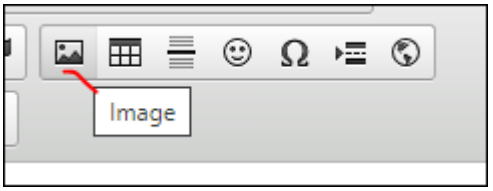
Note that you will need to adjust this code to match string that you are encoding. In this example the code is designed to locate the string *usr=* and to extract the data following the equals sign.

DBMail uses the *Content ID* or “cid” to send images and other files. If you want an image in the email template then proceed as follows:

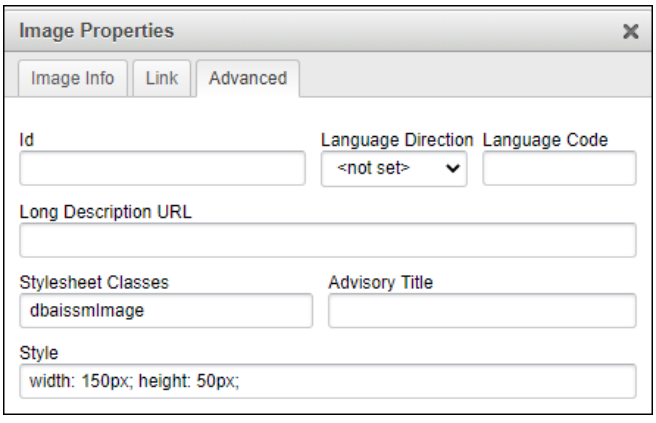
- Open the HTML Editor by clicking the *Edit* button.
- Focus on the point at which you want to place the image.
- In the example below we want to place the image in the cell marked with yellow.



- Click the image button (see image to the right).



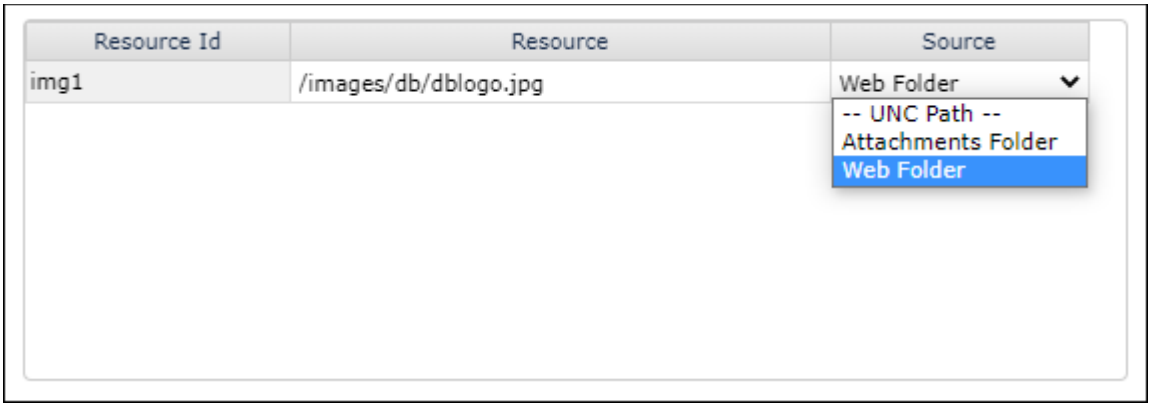
- In the URL field enter “cid:” followed by a unique name for the image; “img1” in our example.



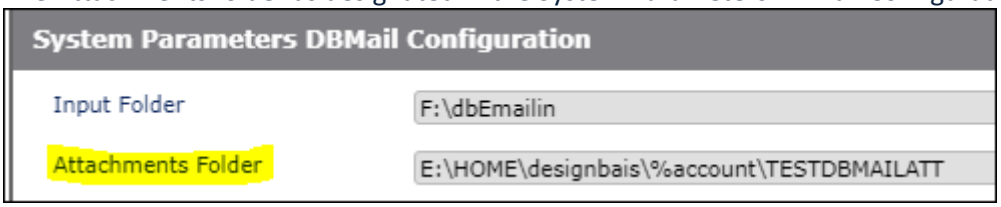
- In the Advanced tab you can enter a stylesheet class and style properties if required.
- If you click the “Source” button in the HTML editor you can view the html source that is produced.

```
<div>
<table border="0">
    <tbody>
        <tr>
            <td style="text-align: center"><img alt="" class="dbaissmImage" src="cid:img1" /></td>
            <td class="dbaissmItem"> </td>
        </tr>
        <tr>
            <td> </td>
            <td>
            <h1><span style="font-family:Calibri,sans-serif;"><span style="font-size:16pt;"><span st
```

- Submit the DBMail template and then recall it. Note that the cid name is extracted into the *Resource Id* grid.



- You can then enter in the *Resource* column the name of the image that you want to link to this cid.
- In the *Source* column select the required source designation:
  - ❖ UNC Path – full universal naming convention path
  - ❖ The Attachments folder as designated in the System Parameters DBMail Configuration



  - ❖ The Web Folder, for example if the DesignBais website is on C drive and named “DBNET” and the name in the *Resource* column is */images/db/dblogo.jpg* then DBMail expects that the image will be found in *C:\DBNET\images\db\dblogo.jpg*

- When using “cid:” to insert an image, the image will ususally not be able to be located. This means that the position of the the image cannot be seen after exiting the HTML editor. To overcome this use the “alt” tag to add either text or simply a space character. The image will then display as a default icon.

<td style="text-align: center"></td>

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

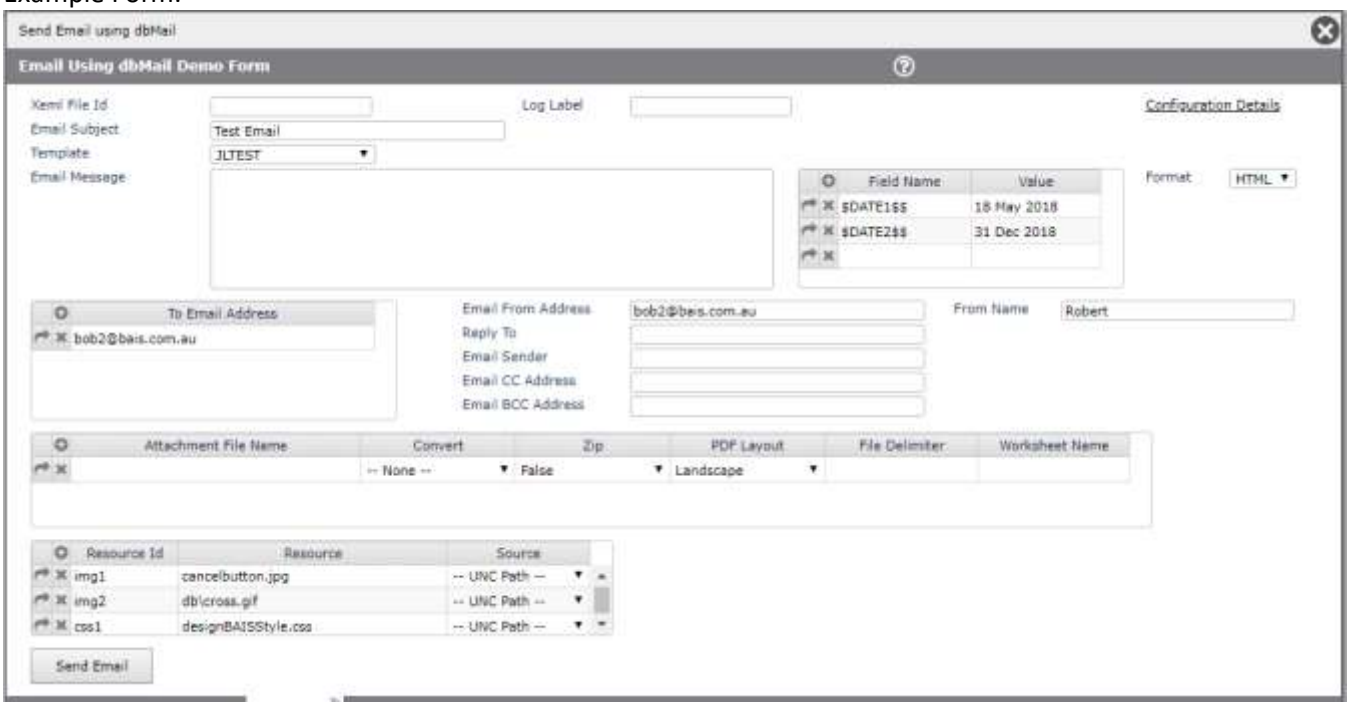
DBMail Demonstration Form

The DesignBais Demonstration Form DBDEMO_DBMAIL can be viewed in the DBINET.DEMO account. It will provide a guide to implementing DBMail.

cid (Content-ID) can be used to send media via email. Attach the image to the email and reference it with HTML tags in the email's template. This embeds the image when it's opened.

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <link rel="stylesheet" type="text/css" href='cid:css1'>
   <title>jl test</title>
</head>
<body>
   <div>
      <table>
         <tr>
            <td style='text-align: center'> </td>
            <td class='dbaissmItem'>
               <img width='183' height='67' class='dbaissmImage' src='cid:img1' /></td>
         </tr>
         <tr>
            <td><img src='cid:img2' /></td>
            <td>This looks like it may be working!</td>
         </tr>
<tr>
            <td>>$$DATE1$$</td>
            <td>Dates in column 1, text in column 2</td>
         </tr>
<tr>
            <td>$$DATE2$$</td>
            <td>Day 2 text</td>
         </tr>
      </table>
   </div>
</body>
</html>
```

Example Form:



The list of *Resource Id* entries correspond to the *cid* entries in the HTML. These are assigned to the DBM.RESOURCE.ID equate name in the dynamic array passed to DBI.G.DBMAIL. The *Field Name* and *Value* ($$DATE1$$) provides a means of replacing placeholders with text specific to this instance of a general email.

The compression of PDF conversion of attachments can be set by adding an entry in the *dbmail.config* file like:

<HIQImageCompressionLevel>85</HIQImageCompressionLevel>

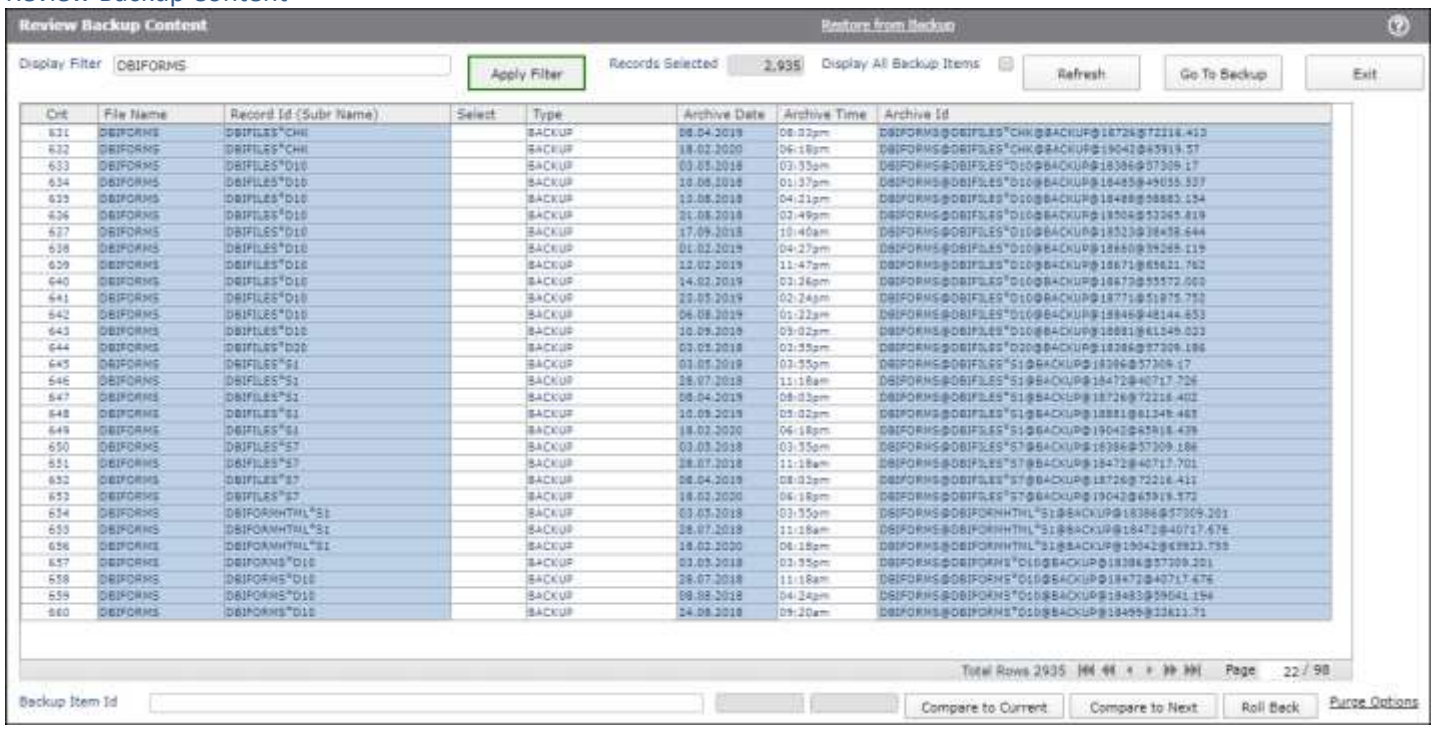
In this example the value of 85 indicates a level of compression that significantly reduces the size of the PDF file without a serious drop in resolution. A lower number will increase the size and resolution. A value of 0 indicates no compression.

Backup Changed Items

This feature is available from the *Backup* button on the *Code Editor* form. Items from selected files are copied to a backup file if the current version of the item is different to the most recent verion on the backup file. This allows multiple versions of items to be retained. There is the ability to compare a backup item to the current version of that item to highlight changes that have occurred since the backup item was generated. There is the ability to roll back an item from backup to overwrite the current version. There are purge options.

There are two main functions.

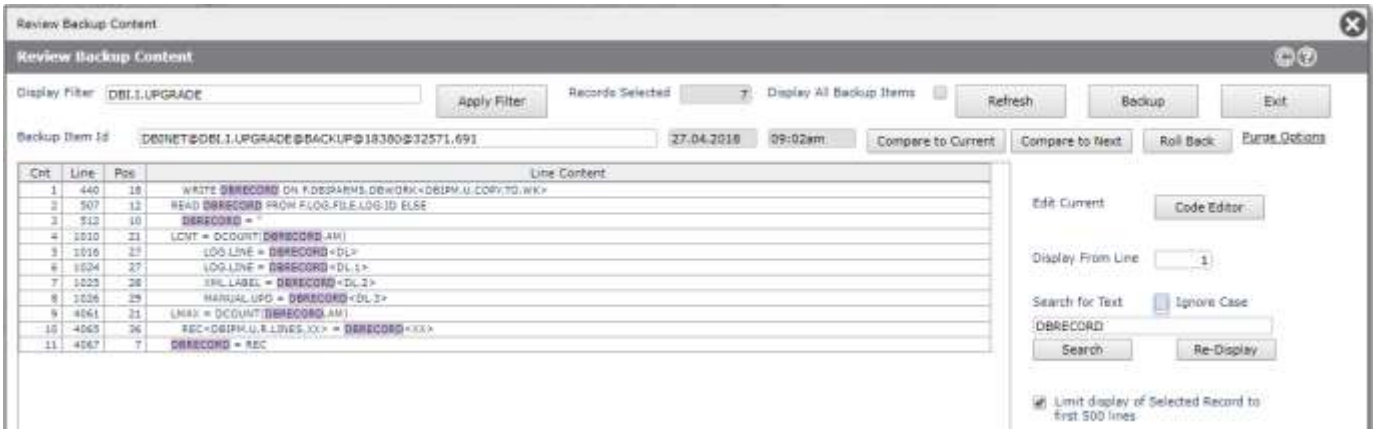
Review Backup Content



Use this form to view backup items. Enter a search string in the *Display Filter* field to limit the display to those items that contain the string.

Click on the *Record Id (Subr Name)* column in the required row to display the backup item. Use the *Code Editor* button to view the backup item in the Code Editor.

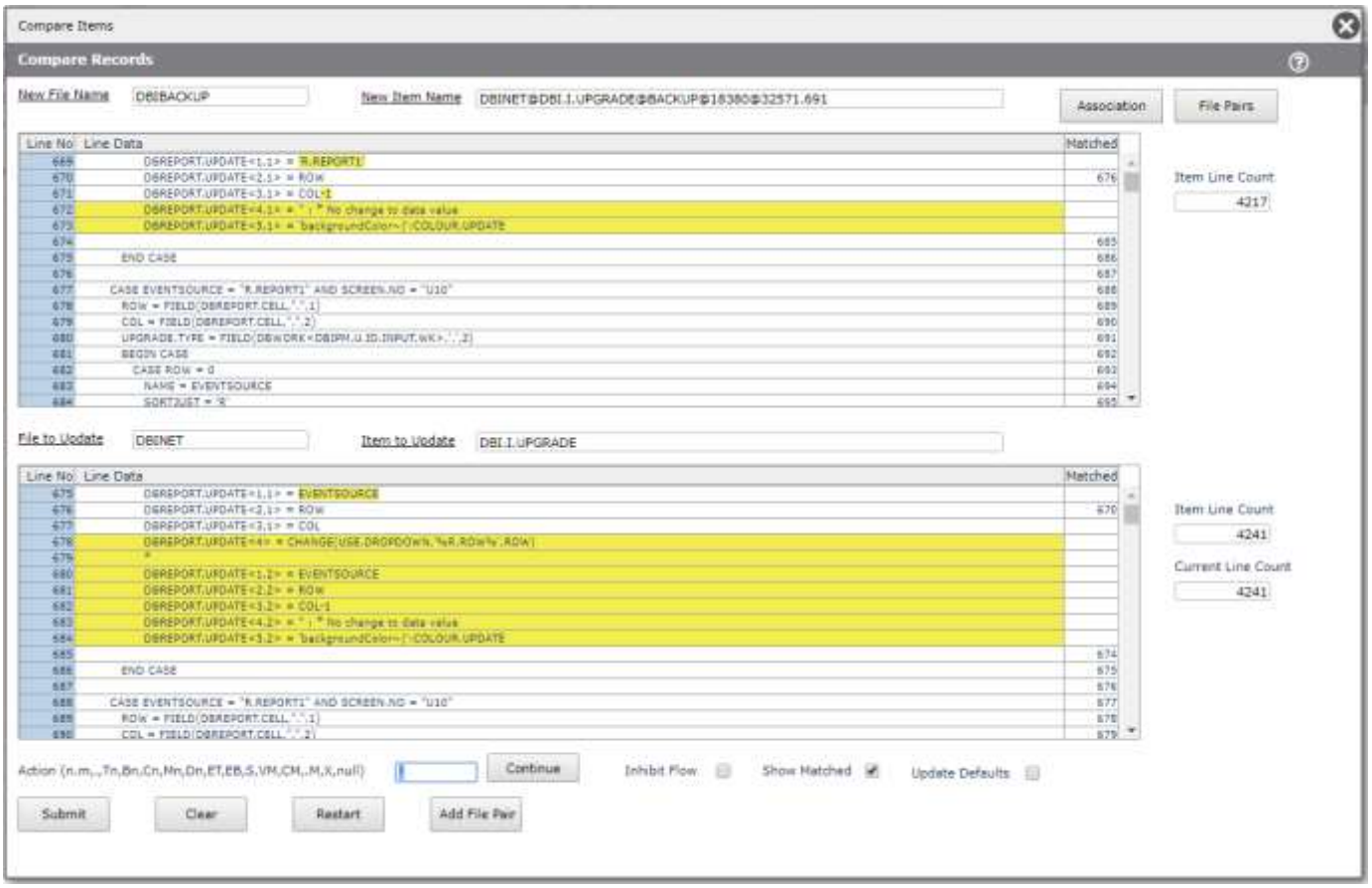
Alternatively the item is displayed as an on-from report. Enter a string in the *Search for Text* field to display lines containing this string. In the example below all lines from a basic subroutine that contain the string *DBRECORD* are displayed.



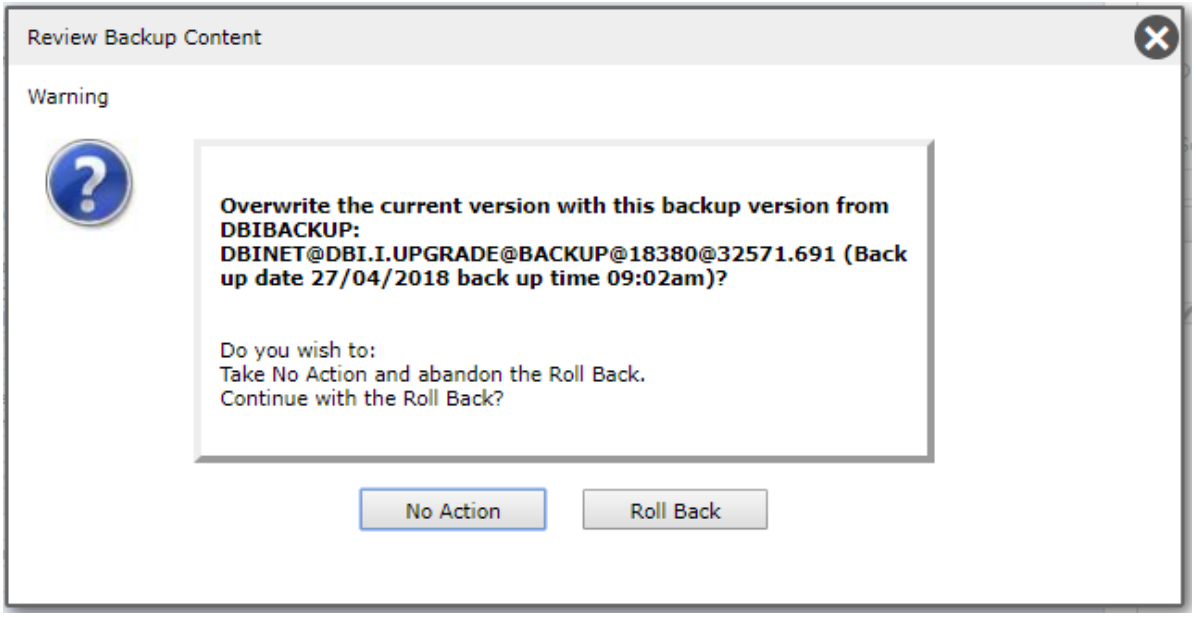
The *Purge Options* button allows for purging of a single item or a range of items that match particular criteria.

The *Compare to Current* button can then be used to invoke the *Compare Items* form which displays the comparison between the backup item and the current version of the item, highlighting differences.

The *Compare to Next* button can then be used to invoke the *Compare Items* form which displays the comparison between the selected backup item and the next most recent version of this same item in the backup file, highlighting differences.

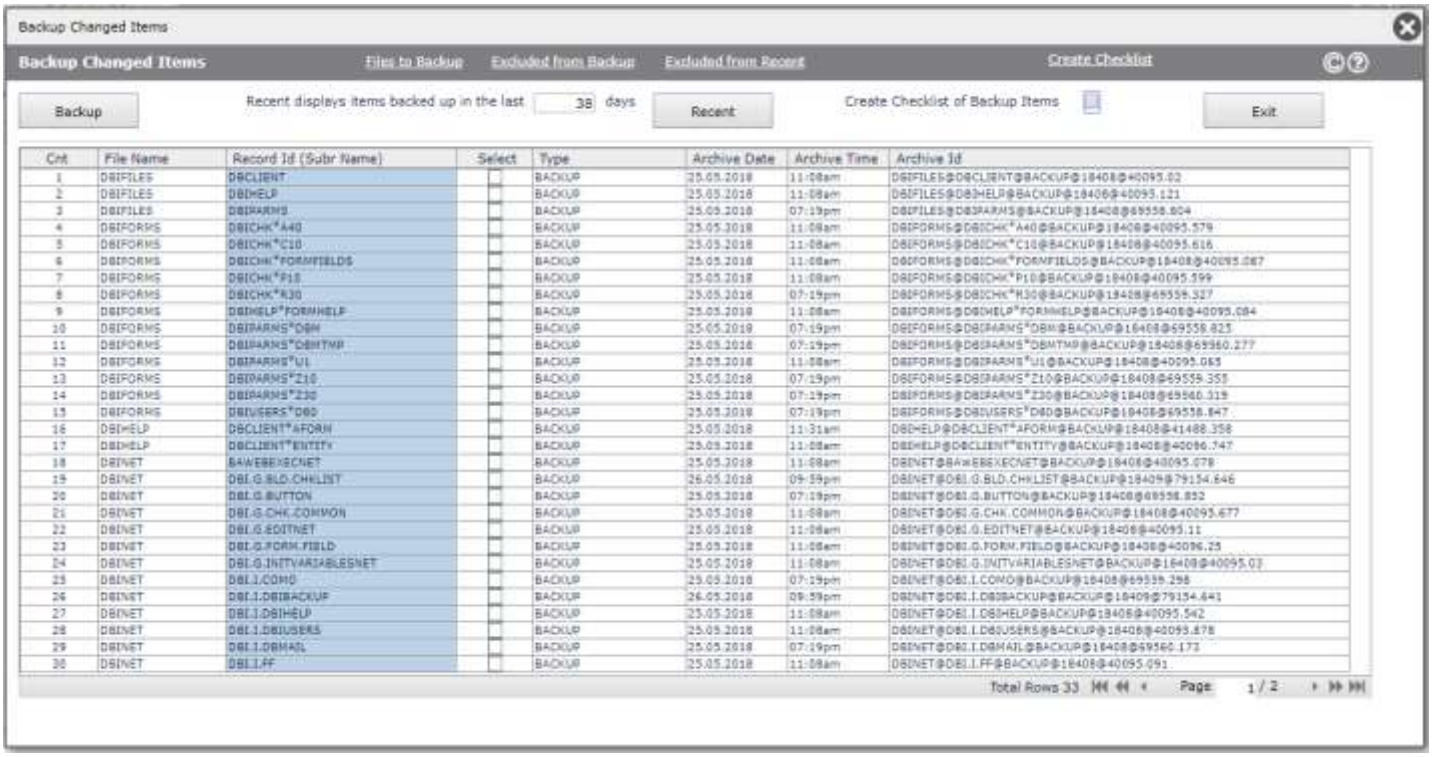


Clicking the *Roll Back* button displays the following form allowing the backup version to be rolled back as the current version. The current version is loaded into the backup file prior to the rollback.



Click the *Go to Backup* button to move to the other main function which is the actual backing up of changed items.

Backup Changed Items



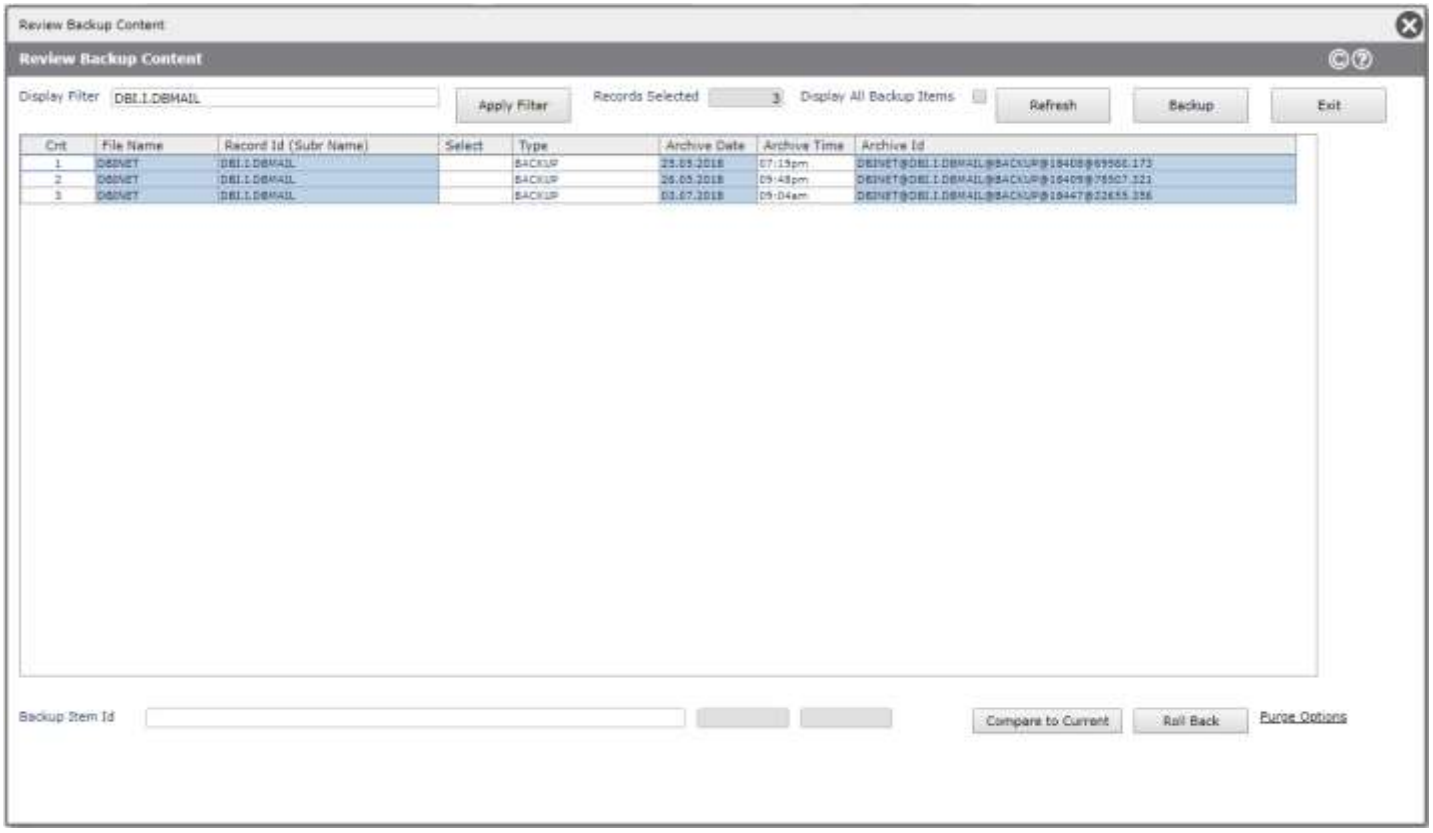
Clicking the *Recent* button displays the list of Recent backup items. The default is to display items backed up in the last 5 days.

Clicking the *Backup* button initiates a pass through all items that are in the DBIBACKUP file to check if the current version differs from the most recently backed up version. Where there is a difference the current version is added to

the backup file. All items in the files to be backed up that have never been backed up (new items) are also added to the backup file.
The form displays the list of items that have been backed up. The *Select* column can be used to flag items that are to be included in a checklist. Click the *Create Checklist of Backup Items* button to flag all displayed items for inclusion in a checklist. Items can also be individually selected or de-selected.

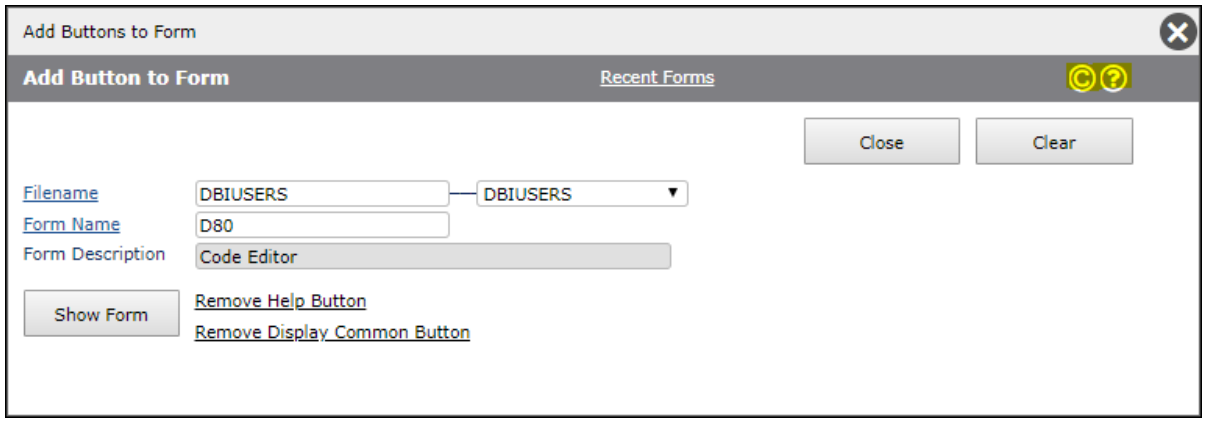
Use the *Create Checklist* button to generate the checklist after selecting the required items.

Click an item in the *Record Id (Subr Name)* column to display all versions of this item from the backup file. In this example there are 3 versions of the routine DBI.I.DBMAIL. Click a version to view the item and compare this version with the current version.



Add Buttons

The *Add Buttons* button on the right hand side of the *Code Editor* form provides a mechanism to add two specific buttons to any form. These button are shown in yellow in the image below.



The button that appears as a copyright symbol is used to display common variables. The circled query mark is used to display form help.

The *Recent Forms* link displays a list of the last ten forms that have been maintained in Forms Designer. Select from this list, or enter the file name and form name, to access the form to which you wish to add either of the two buttons.

Add Help Button to Form
: Click this link to add the form help button to a form (the question mark within the circle).

Remove Help Button
: Click this link to remove the form help button.

Add Display Common Button to Form
: Click this link to add the display common button to a form (the copyright sign).

Remove Display Common Button
: Click this link to remove the display common button.

Click the *ShowForm* button to display the form as a layered enquiry form. This feature is provided merely as a convenient way to check that the buttons have been added or removed, and where they appear in the top of the form.

Google Analytics Tracking

Incorporating Google Analytics into a DesignBais Release 7/8 website can be achieved as described below.

Create a Google developer account such as yourname@gmail.com with password.

Enable Google Analytics for the required website by navigating to: https://analytics.google.com for the above url. Google will provide the JavaSscript code shown below.

Amend the JavaSscript code provided by Google and insert it into the *custom.js* file for the required website. Note that '*UA-99999999-9*' must be replaced with the specific tracking code relating to your website.

```
$(document).ready(function(){
    window.dataLayer = window.dataLayer || [];
    function gtag(){dataLayer.push(arguments);}
    gtag('js', new Date());
    gtag('config', 'UA-99999999-9');
});
```

Insert the Google script reference provided by Google into website *htmlMetaTags.txt* file:
<script async src="https://www.googletagmanager.com/gtag/js?id=*UA-99999999-9*"></script>

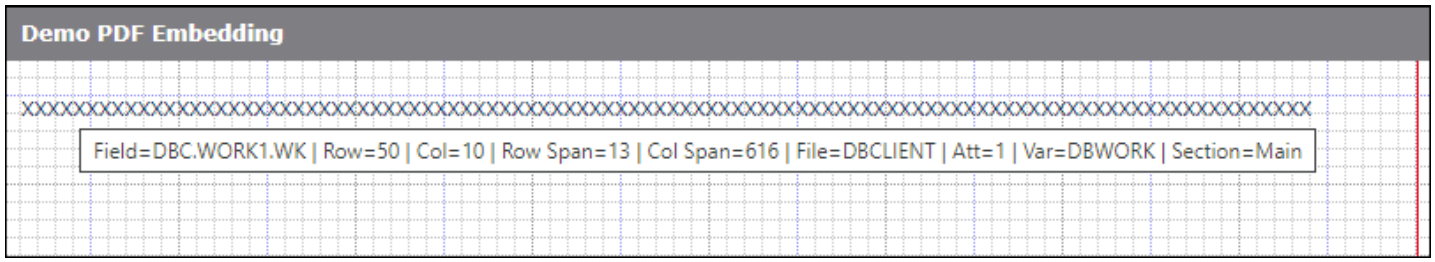
This procedure can be followed to set up Google Analytics for any DesignBais V7 website page.

Global Site Tag (gtag.js)
This is the Global Site Tag (gtag.js) tracking code for this property. Copy and paste this code as the first item into the <HEAD> of every web page that you want to track. If you already have a Global Site Tag on your page, simply add the config line from the snippet below to your existing Global Site Tag.

```
<!-- Global site tag (gtag.js) - Google Analytics -->
<script async src="https://www.googletagmanager.com/gtag/js?id= UA-99999999-9"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'UA-99999999-9');
</script>
```

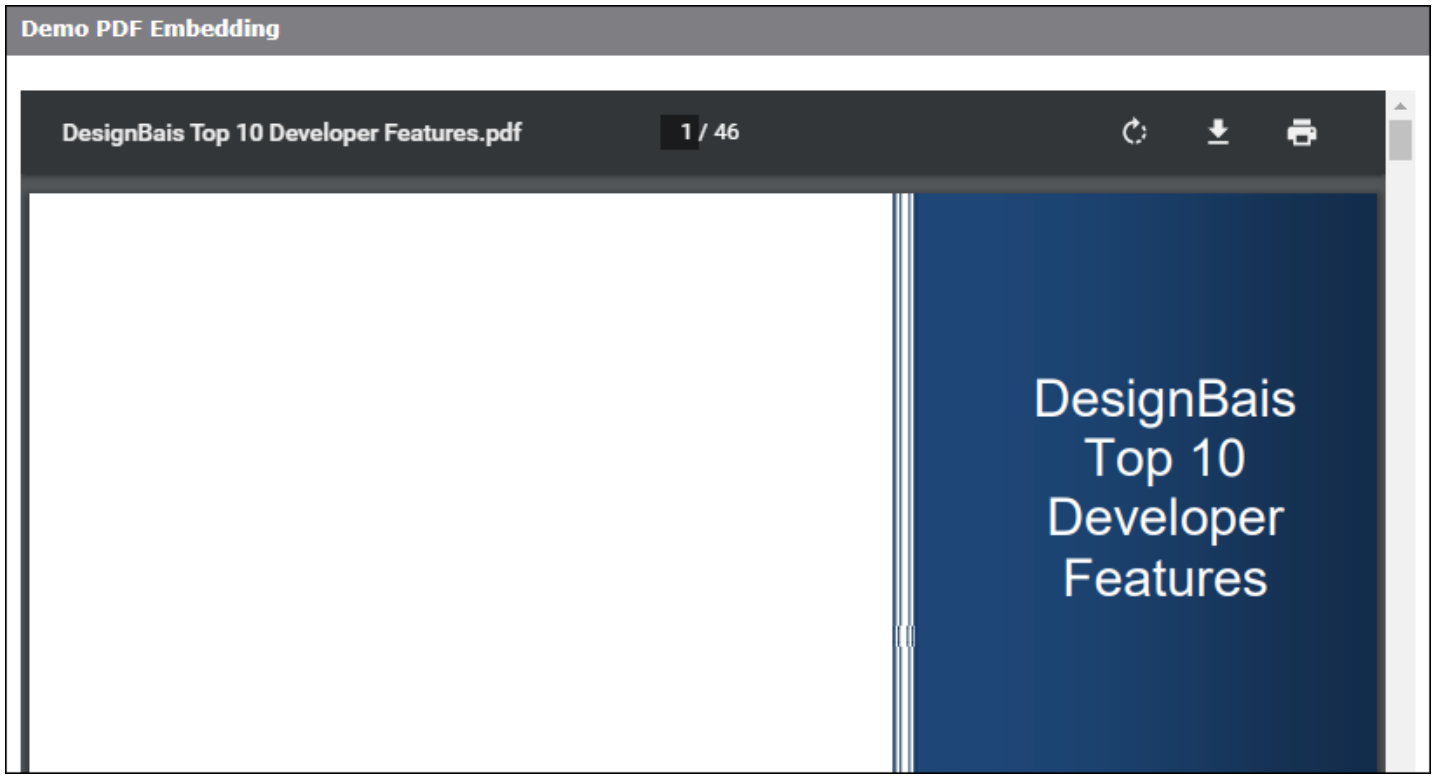
Embedding a PDF in a Form

Simply add an output work field to your form. In the following example the field is called DBC.WORK1.WK.



In the *AFTER DISPLAY* EVENT, or in a validation or button click event, update the work field using the *embed* tag with the name of the pdf document as shown in the following example:

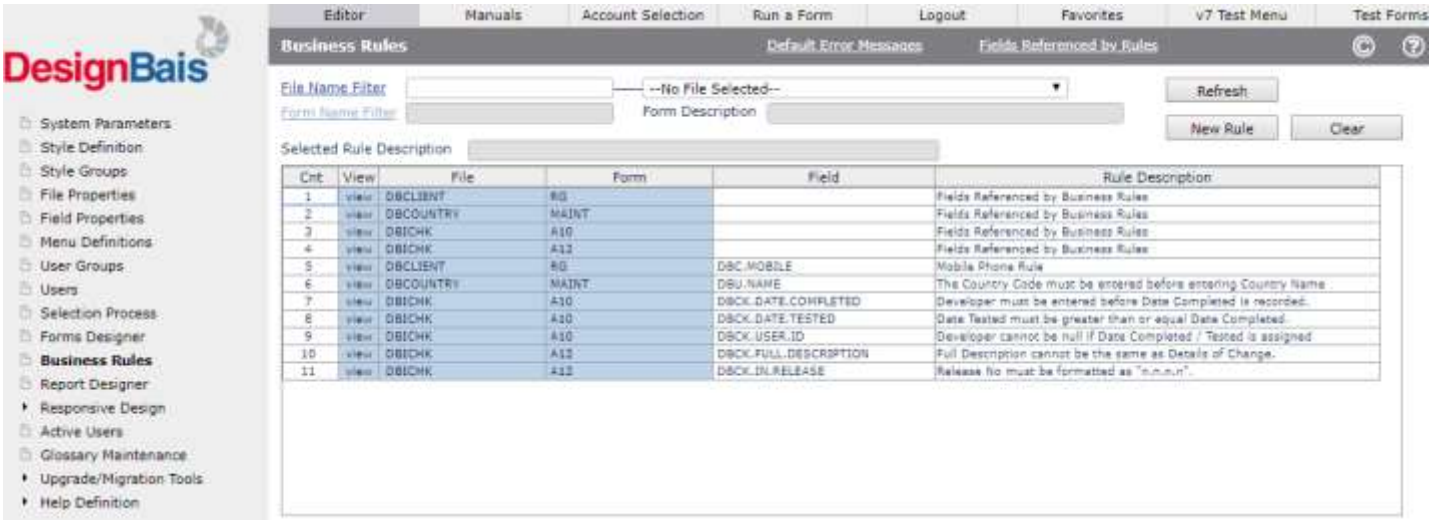
```
DBWORK<DBC.WORK1.WK> = '<embed src="http://www.DesignBais.com/downloads/manual/your_doc.pdf" width=800 height=1100 />'
```



DesignBais Business Rules

It is normal for developers to build business rules into the forms and basic code that comprise applications. Business Rules provides System Administrators or Users the means to implement form field validation rules without the need to involve programmers.

The Business Rules link is on the side menu of the Developer Tools form. This option can be added to user menus or favorites lists if access is to be provided to those without access to the DesignBais tools.



File Name Filter An entry in this field will filter the display of rules to those that are on the nominated file. Click the Refresh button to re-display the report.

Form Name Filter If populated then rules pertaining to the nominated form name will be displayed.

View Click the row in the View column to view or maintain a business rule.

Refresh Click to refresh the display based on the values in the filter fields.

New Rule Click to create a new rule using the *Form Field Business Rules* form.

Default Error Messages Display and maintain standard default error messages.

Fields Referenced by Rules

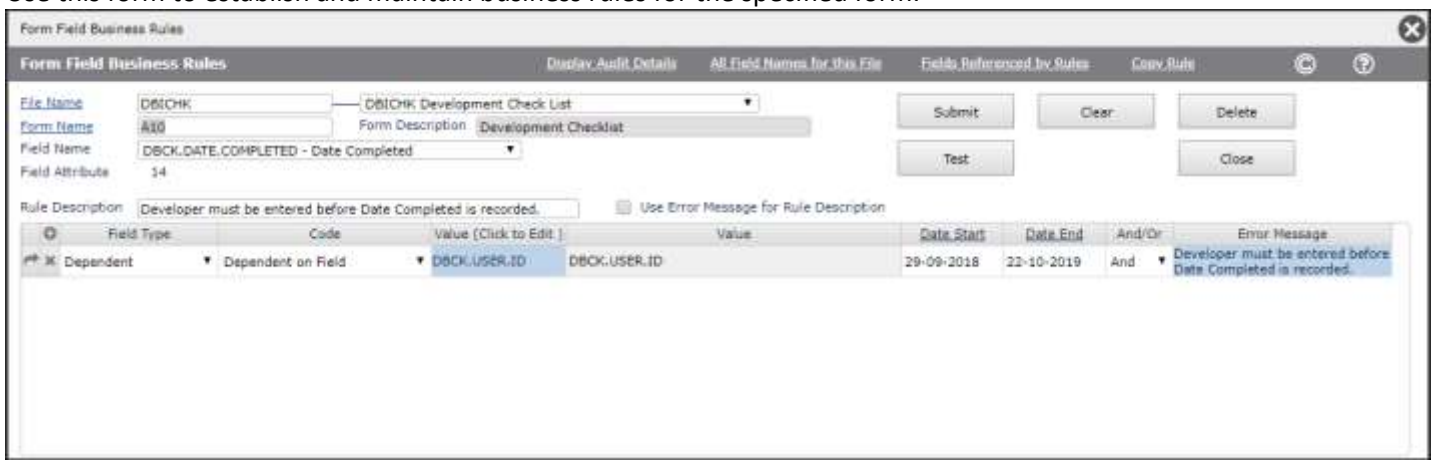
Displays a list of all fields on the form that are referenced by any of the rules that exist for the form. This list of fields does not include the primary fields that the rule applies to, unless those primary fields are also referenced fields in other rules.

This list is maintained for display purposes and is also used in Forms Designer to prevent the deletion of form fields that have a business rule, or are referenced by a business rule. The business rule applying to a form field must be deleted before the form field itself can be deleted.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Form Field Business Rules form

Use this form to establish and maintain business rules for the specified form.



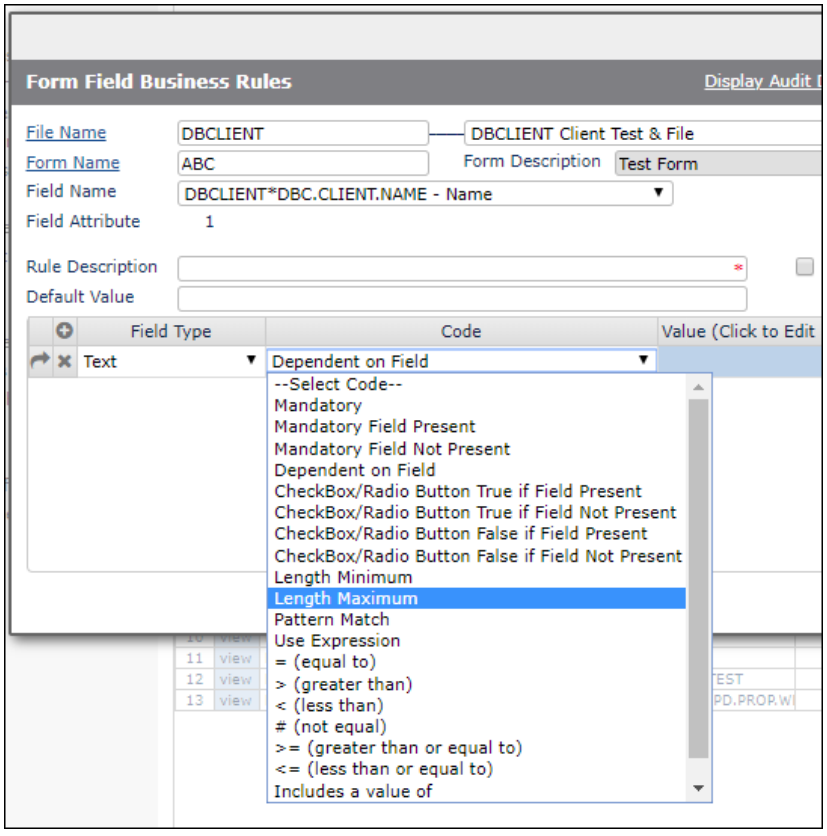
File Name The file name of the form to which the rule is attached.

Form Name The name of the form.

Field Name Select the field name to which the rule will apply.

Field Type Select from the dropdown list.

Code Select from the dropdown list. See below.



Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Select the applicable code from the dropdown list.
[In the help text below the field or value entered in the next column is referred to as the Referenced Field/Value]

Mandatory - the rule field must have a value

Mandatory Field Present - the rule field must have a value if the Referenced Field has a value.

Mandatory Field Not Present - the rule field must have a value if the Referenced Field has no value.

Dependent on Field – the Referenced Field must be populated BEFORE the rule field can be entered.

Check Box/Radio Button True if Field Present - If referenced field is populated then DBC.CLIENT.NAME check box must be set to true.

Check Box/Radio Button True if Field Not Present - If referenced field is NOT populated then DBC.CLIENT.NAME check box must be set to true.

Check Box/Radio Button False if Field Present - If referenced field is populated then DBC.CLIENT.NAME check box must be set to false.

Check Box/Radio Button False if Field Not Present - If referenced field is NOT populated then DBC.CLIENT.NAME check box must be set to false.

Length Minimum – the length of the rule field value must be greater than or equal to the Referenced Value.
Length Maximum – the length of the rule field value must be less than or equal to the Referenced Value.

Pattern Match – the value in the rule field must match the pattern specified in the next column [use 0X, 3X, 0N, 4N type nomenclature]

Use Expression – enter an expression in the next column.

Arithmetic symbols - For example select '< (less than)' in this list and then enter the Referenced Value. The rule will then become 'If Rule Field < Referenced Value'.

Includes a value of - the rule will check that the field contains a value that includes the Referenced Value.

Value Click the blue highlighted column to open the *Business Rule Value Entry* form. The display of this form is context sensitive based on the value entered into *Field Type* and *Code*.

Date Start The date from which the business rule will be invoked. Enter a future date if you want to set up rules in preparation for system implementation.

Date End The date after which the business rule will cease to be active. Use this field to remove the effect of a rule without deleting the rule. It can then be re-invoked if required.

Error Message The error message to display if data to which the rule applies does not satisfy the rule conditions. If a standard default error message is used then merely enter the *Error Message Id* of the assigned default message.

Display Audit Details Displays when and by whom the error messages were maintained.

All Field Names for this File
Displays, for reference, the Field Properties for the file to which the rule will apply.

Fields Referenced by Rules
Displays a list of all fields on the form that are referenced by any of the rules that exist for the form. See above.

Copy Rule Allows a rule to be copied to another field or another file. See below for details.

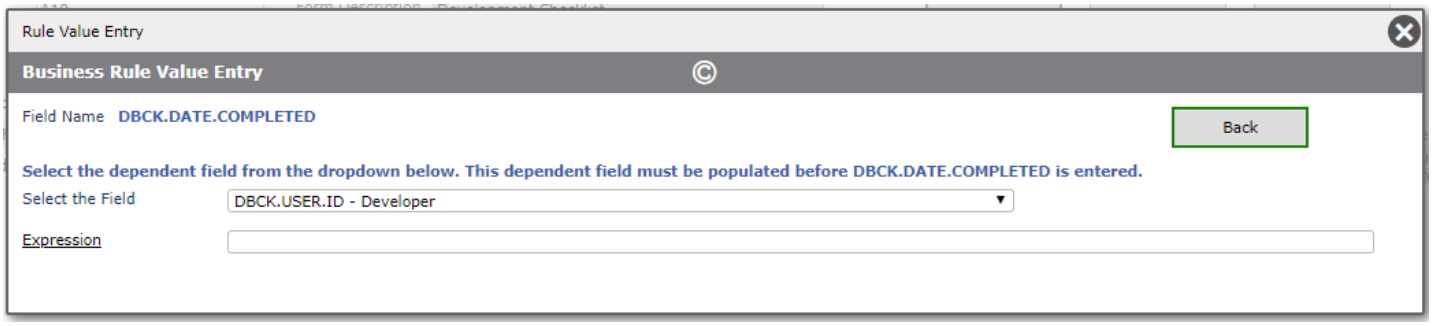
Submit Save the rule definition on the DBIRULES file. The key for the rule record is formed by concatenating *FileName*FormName*FieldName*.

Test Allows the rule logic to be tested against selected records on the file.

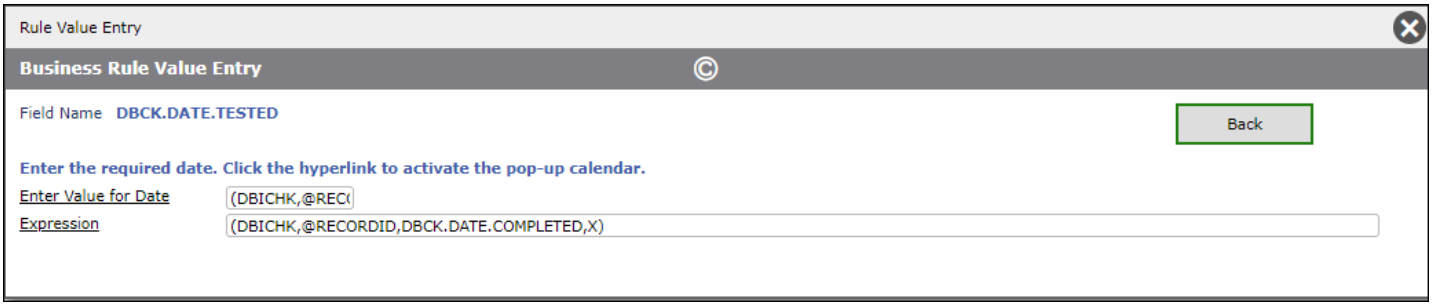
After selecting a *Field Name* from the dropdown list the *Rule Value Entry* form is displayed.

If the rule is to reference another field on the form then select the field from the dropdown.
If the rule requires a value to be entered then:

- either use the displayed entry field which will be of a type that matches the type of the rule field
- or click the *Expression* button to open the *Business Rules Define Expression* form



An example of an expression is shown below. Expressions can be entered freehand if the syntax is known otherwise use the *Business Rules Define Expression* form options.



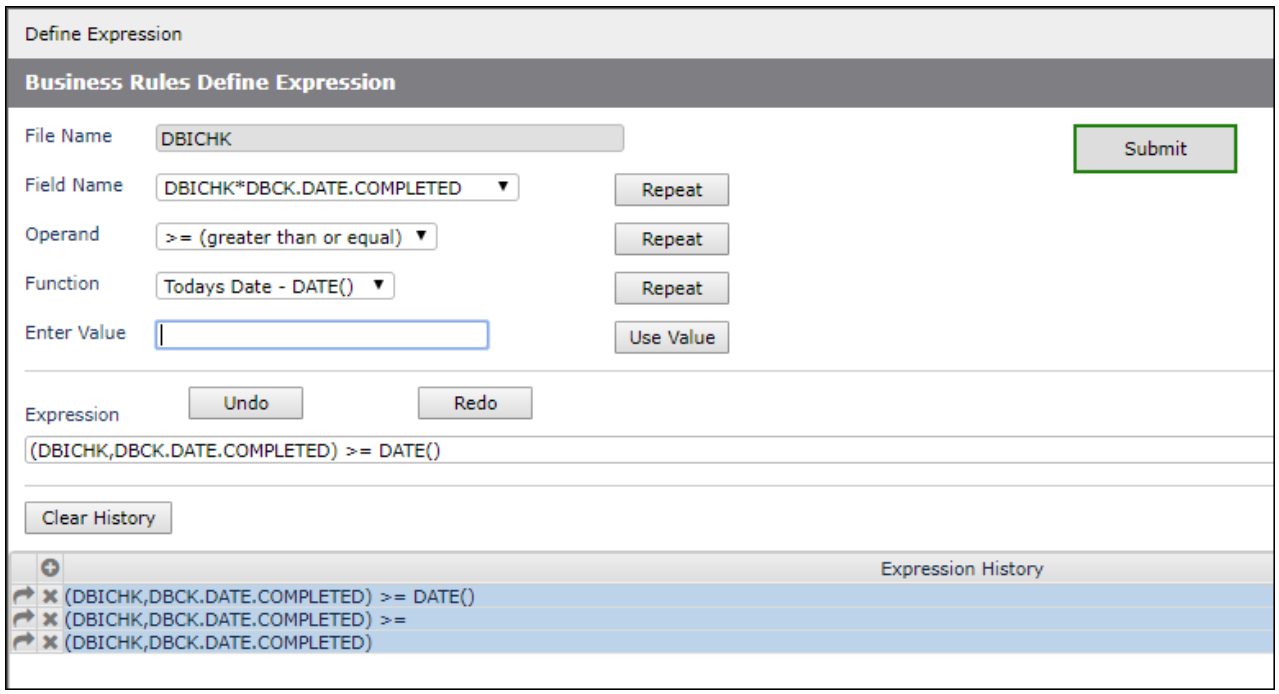
Click the Expression link to open the *Business Rules Define Expression* form.

Select any of the options in order to create the required expression.

- Selecting a field name from the Field Name dropdown will create the required expression to access the value in that field.
- Selecting an Operand or a Function will insert that code into the expression.
- Use the *Repeat* button to use the same code again.

In the following example an expression has been created to provide for the business rule condition:

- DBCK.DATE.COMPLETED must be greater than or equal to Todays Date



The *Test* button provides a means of testing a rule from within the *Form Field Business Rules* form.

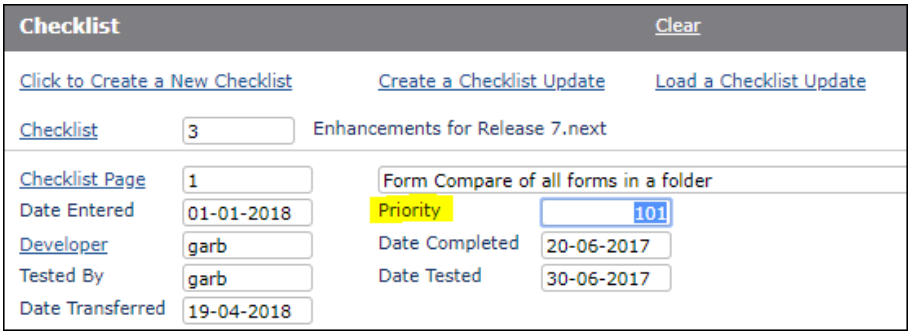
Clicking the *Test* button reveals the *Rule Test Result* fields.

Select a *Record Id* from the dropdown list, which contains a sample of records from the file to which the rule is attached. If necessary the relevant field or fields on one of the sample records can be edited such that the rule outcome is either a pass or a fail.

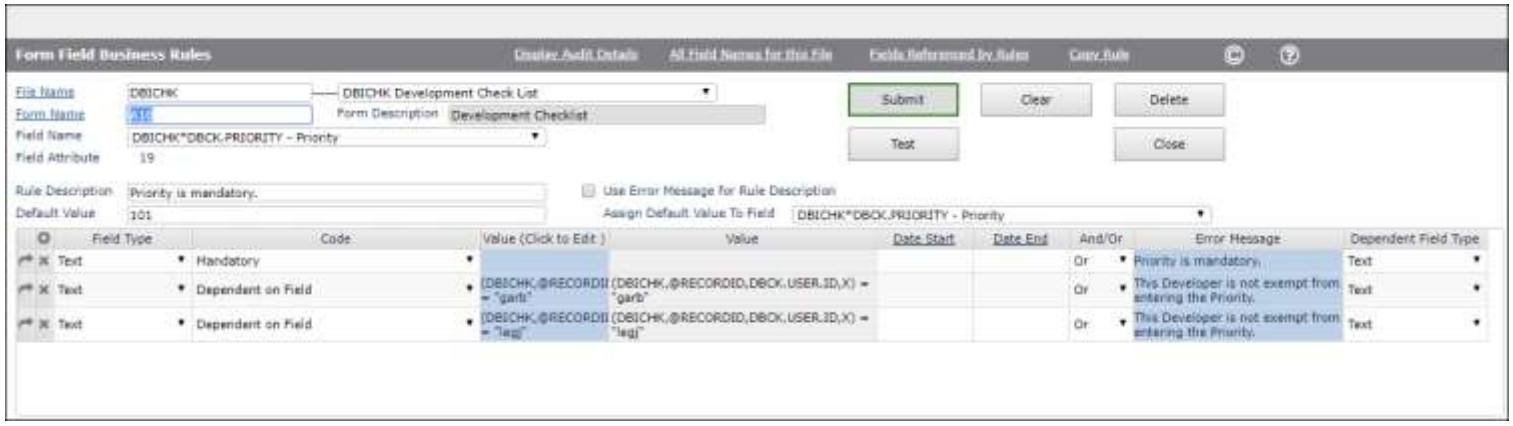
The reults of the rule evaluation are displayed. In the example below there are 2 conditions for the rule, joined by the *OR* conjunctive. Hence the *Pass Status* is **Pass**, even though 1 of the conditions failed to pass.

Example Business Rule demonstrating the use of an expression in order to suppress the display of the error message for certain users.

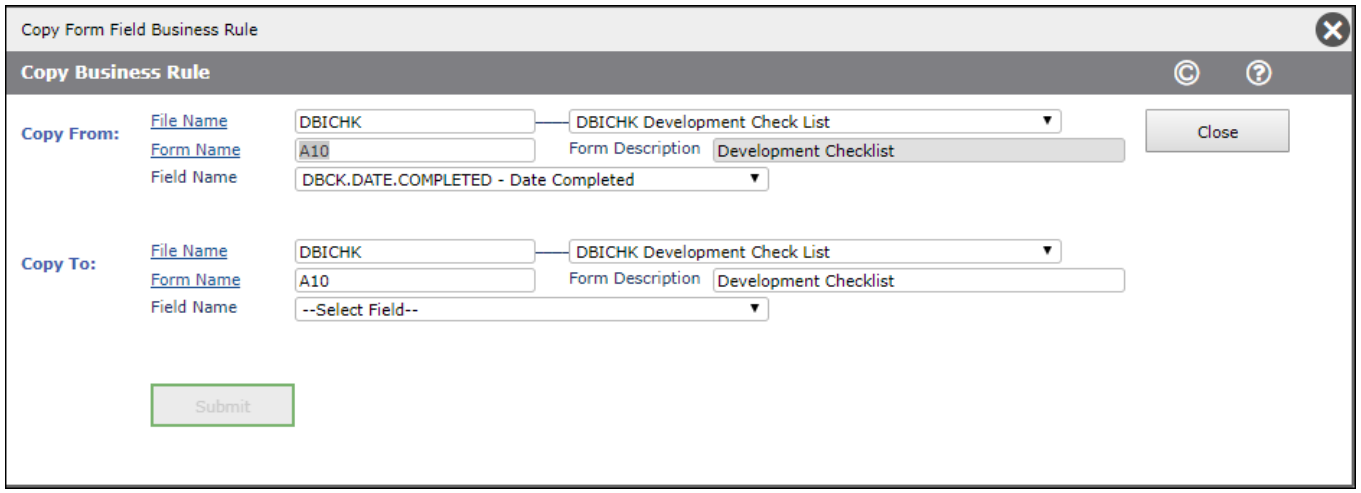
In this example, from the DesignBais Development Checklist function, the business rule is set to enforce the entry of the priority for the checklist task. If priority is not entered then the message "*Priority is mandatory*" displays after clicking Submit. The default value of 101 is loaded into the priority field and when Submit is clicked again the record is saved.



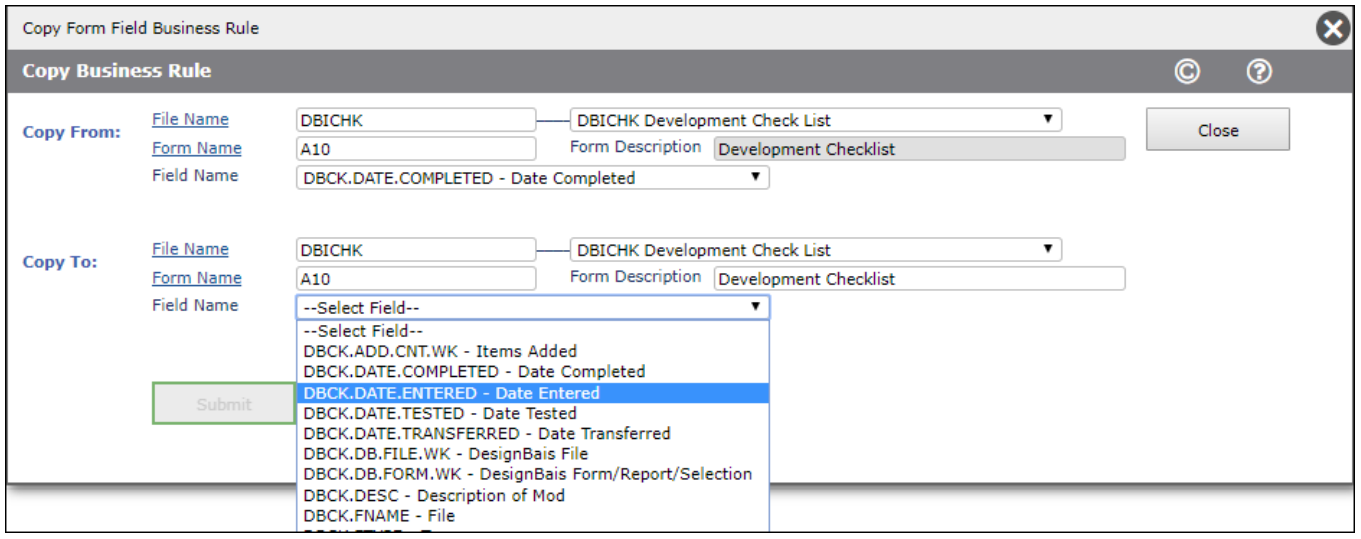
If you want to suppress the display of the message for selected users but still retain the action of the rule then set up additional lines with an expression as shown below. In this example the expression evaluates the content of the DBCK.USER.ID field and compares this value to the string "garb" or "legj". The result of the expression is either true (value of 1) or false (value of 0). Setting the *And/Or* value to "Or" allows any of the 3 lines to signal a that the rule validation has been passed. For users "garb" or "legj" the message is suppressed but the default action takes place anyway.



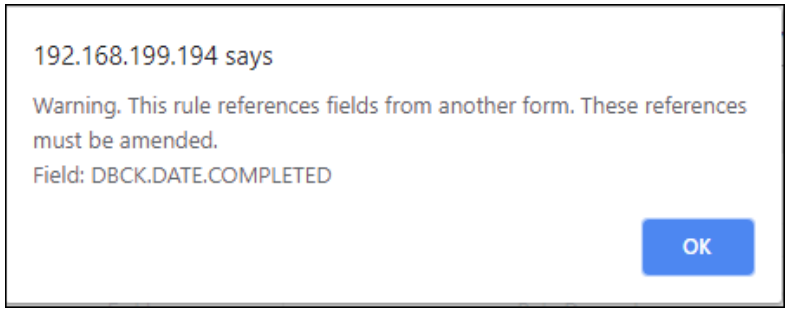
Copy Business Rule form.



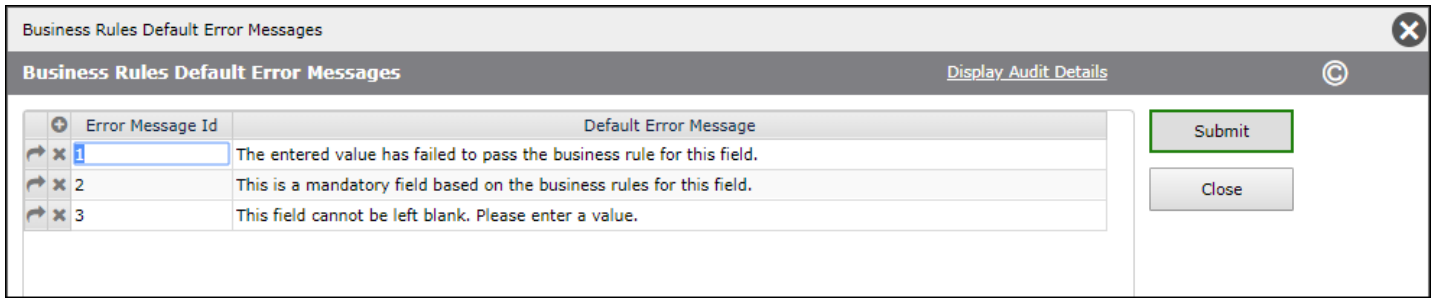
Copy To Select a file and form, or accept the defaults and select another field on the current form.



Note that the rule that is being copied may reference other fields on the form. If, therefore, you wish to copy this rule to another form then the referenced fields may not be on the other form. A warning message, see the example below, will display but the copy of the rule is permitted.



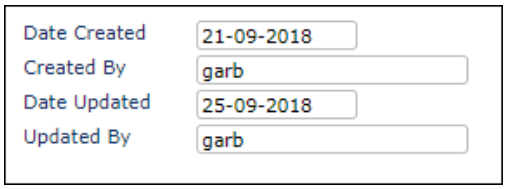
Business Rules Default Error Messages form.



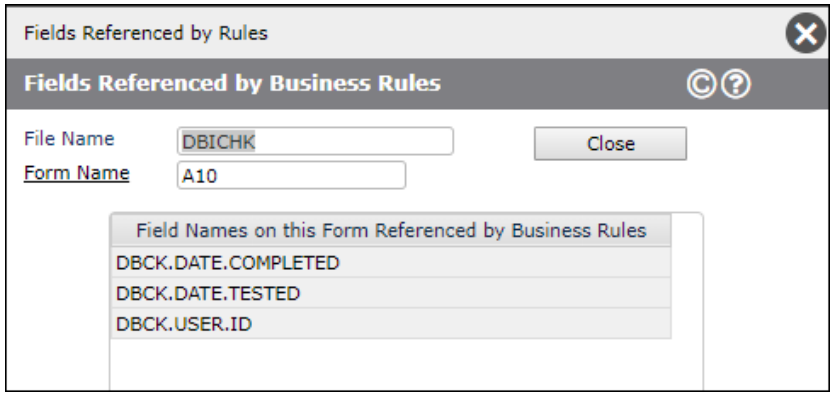
Error Message Id Assign a code as a handle for each message. A short code makes sense, as shown above where default messages are assigned integer ids.

Default Error Message The text of the error message.

Display Audit Details Displays when and by whom the error messages were maintained.



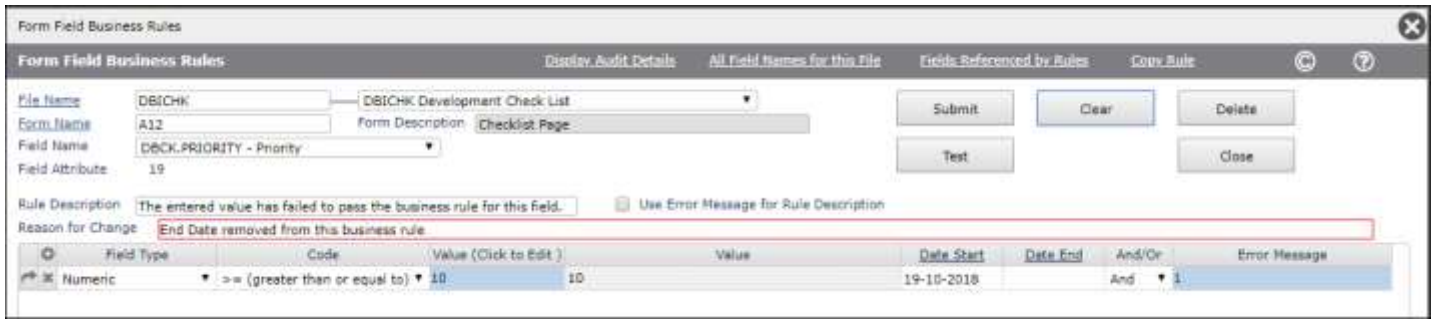
Fields Referenced by Rules form



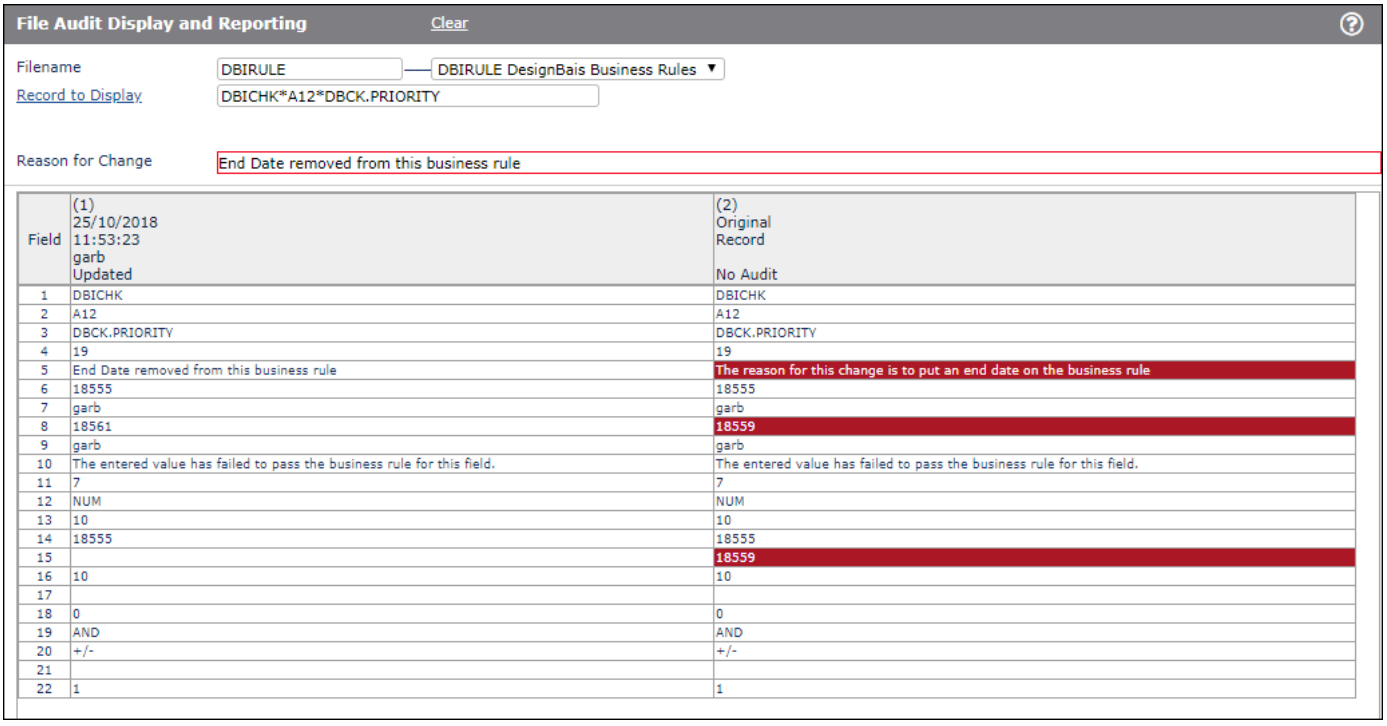
Audit of changes to Business Rules

The DBIRULE file is set up to provide for *Extended Audit*. This property is maintained in the File Properties form.

By default any change to a business rule will be recorded. The *Reason for Change* field is displayed and is mandatory.



When viewed in the *Audit Display* function the reason for change of a business rule is highlighted.



Field	(1) 25/10/2018 11:53:23 garb Updated	(2) Original Record No Audit
1	DBICHK	DBICHK
2	A12	A12
3	DBCK.PRIORITY	DBCK.PRIORITY
4	19	19
5	End Date removed from this business rule	The reason for this change is to put an end date on the business rule
6	18555	18555
7	garb	garb
8	18561	18559
9	garb	garb
10	The entered value has failed to pass the business rule for this field.	The entered value has failed to pass the business rule for this field.
11	7	7
12	NUM	NUM
13	10	10
14	18555	18555
15		18559
16	10	10
17		
18	0	0
19	AND	AND
20	+/-	+/-
21		
22	1	1

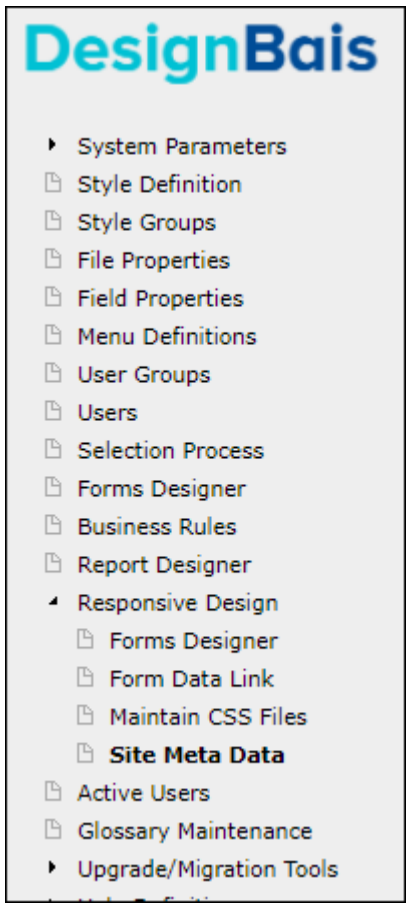
Workflow

The DesignBais Workflow module is included in the DesignBais release software.

For a full explanation of this module refer to the DesignBais Workflow reference manual.

The manual is accessed from the *Manuals* option on the Development Tools top menu which accesses the *Resources* tab on the DesignBais website www.designbais.com.

Responsive Design



DesignBais Release 8 includes the Responsive Design Toolset.

Access is from the DesignBais Developer Tools side menu.

There is a DesignBais Responsive Design reference manual accessible from the *Manuals* top menu option on the Developer Tools form which accesses the *Resources* tab on the DesignBais website www.designbais.com.

Refer to the DesignBais Responsive Design reference manual for a description of the options:

- Form Data Link
- Maintain CSS Files
- Site Meta Data

Developers can call a Responsive Design page from a traditional DesignBais form using the externalDBModal flag on a DBCALLURAL with a page query string parameter will invoke the Responsive Design Page in a separate tab.

The original tab will be paused until you return.

Return by DBCALLURL to your website with return.aspx replacing dbnet.aspx.

Common will be passed when pages are flagged as Preserve Common.

If you want to use Google Analytics in your Responsive Design website to monitor website hits then the following code must be loaded into the *Site Meta Data* form which is accessed from the DesignBais Tools side menu shown above.

Here is the code. Load it as shown in the snips below.

```
<!-- Google Tag Manager -->
        <script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
        new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
        j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
        'https://www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
        })(window,document,'script','dataLayer','GTM-T8BVRBV');</script>
<!-- End Google Tag Manager -->


<!-- Google Tag Manager (noscript) -->
        <noscript><iframe src="https://www.googletagmanager.com/ns.html?id=GTM-T8BVRBV"
        height="0" width="0" style="display:none;visibility:hidden"></iframe></noscript>
<!-- End Google Tag Manager (noscript) -->
```

Responsive Design Site Meta Data

Site: test

CSS:

Script and Meta Tags:

```
<!-- Google Tag Manager -->
<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
'https://www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer','GTM-T8BVRBV');</script>
<!-- End Google Tag Manager -->
```

Body Elements:

```
<!-- Google Tag Manager (noscript) -->
<noscript><iframe src="https://www.googletagmanager.com/ns.html?id=GTM-T8BVRBV"
height="0" width="0" style="display:none;visibility:hidden"></iframe></noscript>
<!-- End Google Tag Manager (noscript) -->
```

It will appear in the CSS as shown below.

```
<link href="res/test/css/default.css?v=4216" rel="stylesheet">
<!-- Google Tag Manager -->
<script>
(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.async=true;j.src=
'https://www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer','GTM-T8BVRBV');
</script>
<!-- End Google Tag Manager -->
<script type="text/javascript" src="https://maps.googleapis.com/maps/api/js?key=AIzaSyD7
defer></script>
<meta name="description" content="DesignBais Development Tools">
<meta name="robots" content="noarchive, nofollow, noindex">
<script type="text/javascript" charset="UTF-8" src="https://maps.googleapis.com/maps-api
<script type="text/javascript" charset="UTF-8" src="https://maps.googleapis.com/maps-api
<script type="text/javascript" charset="UTF-8" src="https://maps.googleapis.com/maps-api
<script type="text/javascript" charset="UTF-8" src="https://maps.googleapis.com/maps-api
</head>
```


<body style="margin-top: 0px; margin-bottom: 50px;">
<!-- Google Tag Manager (noscript) -->
<noscript>
"<iframe src="https://www.googletagmanager.com/ns.html?id=GTM-T8BVRBV"
height="0" width="0" style="display:none;visibility:hidden"></iframe>"
</noscript>
<!-- End Google Tag Manager (noscript) -->
<form version="1" id="mainform" style="display: block;">…</form> == $0

Length of Record Ids

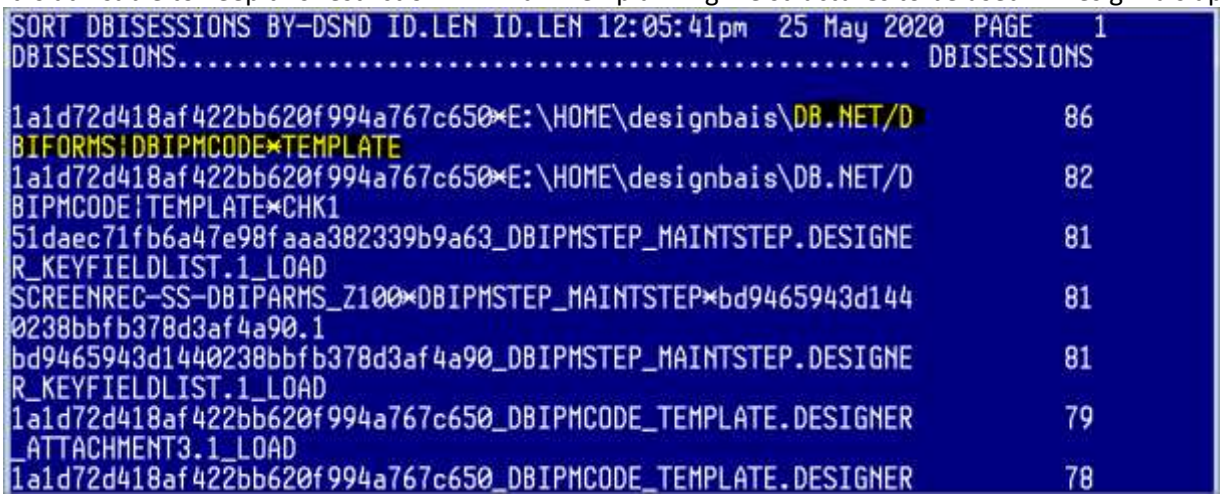
Developers should bear in mind that there is a limit to the allowable length of record ids. The following warning relates to the UniVerse database.

Length of record IDs

Record IDs in UniVerse files must not exceed 255 bytes. This means that the maximum number of characters in a record ID depends on the character set in use. For multibyte character sets, the safe limit is 85 characters. This allows each character to be three bytes long in the internal character set.

This limit also applies to values used as keys in secondary indexes. If a secondary index is too long, a WRITE statement fails, a message is issued, and a nonzero value is returned to the STATUS function.

DesignBais generates keys that may become very long because of the length of the session id.
It is advisable to keep this restriction in mind when planning file structures to be used in DesignBais applications.



Web Service Tester

The test tool supports both TLS 1.1 and 1.2.

Please contact DesignBais for advice on where to download the form:
../downloads/DBServiceTester.exe

UniVerse NLS

On systems where UniVerse NLS is implemented it may be necessary to disable NLS using the following command at the TCL prompt:

SET.FILE.MAP DBISYSFORMS NONE

In this example NLS mode will be disabled for the DBISYSFORMS file.

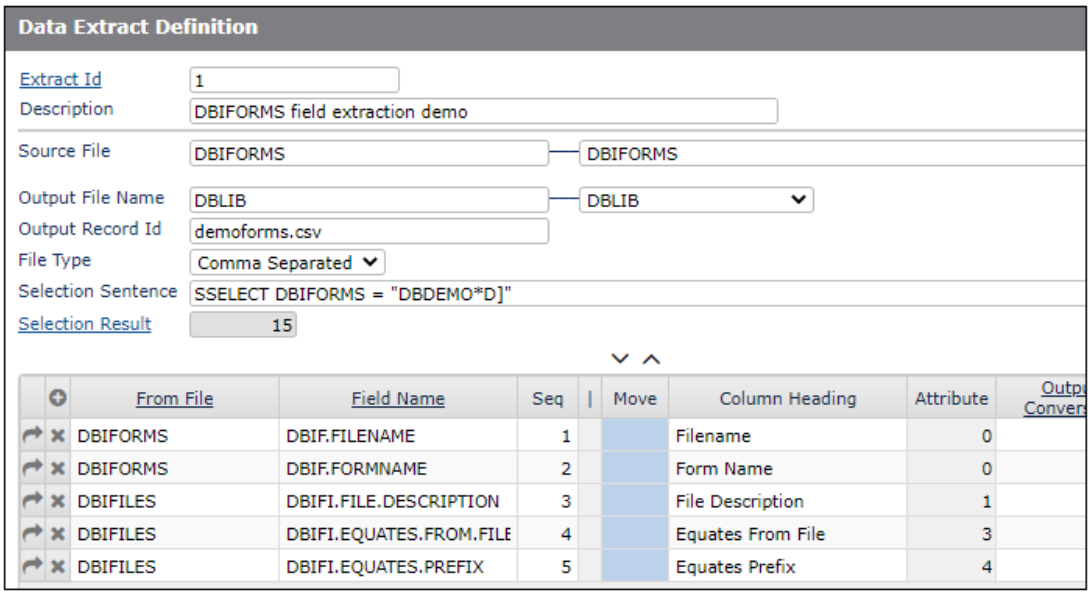
If NLS is not turned off and there is a character in the forms record that is not defined in the NLS map then the form will not display. There may be no error message at all, just a blank form.

DesignBais Data Extraction

Create a flat file containing fields from a selected file and, if desired, fields from any other file that can be linked to the selected file.

- Start by either copying an existing extract definition or creating a new definition.
- Enter the name of the primary file from which the data is to be extracted.
- Specify the name of a file and an output record id in which to create the flat file.

Here is a screen shot of a simple definition:



Data Extract Definition

Extract Id: 1
Description: DBIFORMS field extraction demo
Source File: DBIFORMS — DBIFORMS
Output File Name: DBLIB — DBLIB
Output Record Id: demoforms.csv
File Type: Comma Separated
Selection Sentence: SSELECT DBIFORMS = "DBDEMO*D]"
Selection Result: 15

From File	Field Name	Seq	Move	Column Heading	Attribute	Outp Convers
DBIFORMS	DBIF.FILENAME	1		Filename	0	
DBIFORMS	DBIF.FORMNAME	2		Form Name	0	
DBIFILES	DBIFI.FILE.DESCRIPTION	3		File Description	1	
DBIFILES	DBIFI.EQUATES.FROM.FILE	4		Equates From File	3	
DBIFILES	DBIFI.EQUATES.PREFIX	5		Equates Prefix	4	

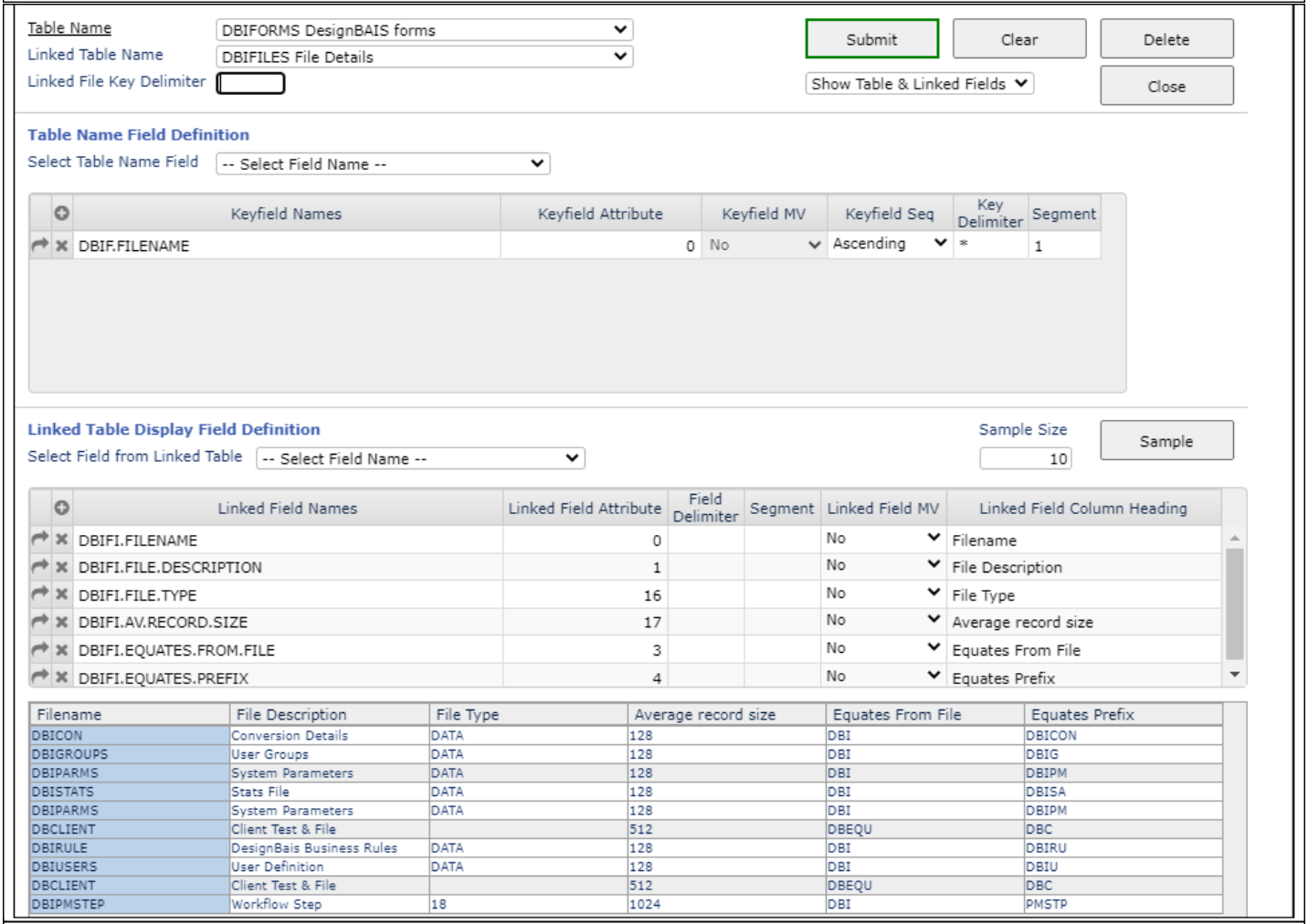
Specify the type of file that is to be created and a selection sentence. This allows you to select specific records is required.
The number of records that will be selected is displays in the selection result field. Click the hyperlink *Selection Result* to refresh the count.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Specify the fields that are to be extracted. There is a lookup *From File* on the grid header which will allow you to access the primary file and any linked files.

Note that fields from DBIFILES can be specified because DBIFILES has been linked to DBIFORMS using the *Table Links* button on the header bar.
A link is possible because the key of a record in DBIFORMS contains the file name which is the key of a record in DBIFILES.

Here is a screen shot of the Table Links record defining this link:



Click the *Run Extract* button to generate the flat file. The details of the run will be displayed in the run log section at the base of the form.

Click *Display Output* to see a sample of the file. You can then enter an email address and send the file to selected recipients.

The email is sent using DBMail. You can select to convert the file to excel or pdf format.

Copyright © 2021 DesignBais Reference Manual Release 9.1.1.3

Field Tag	Help Text	Field Name
©	Displays Developer Tools allowing the developer to view the content of DesignBais common variables at the point of clicking this button.	B.DBICOMMON
?	Displays user help (F1 field help) for all the fields on the form. Help for buttons, images, reports and other form elements can also be defined and displayed.	B.DBIFORMHELP
Copy Extract Definition	Allows you to copy and existing data extract definition to a new id. The new defintion can then be amended as required.	B.COPY
Table Links	Click to display the Table Links maintenance form. If a link from the source file to another file can be defined then the DesignBais Data Extraction can extract fields from both the source file and the linked file.	B.LINKS
View Extract Details	Each time you run an extract definition and create an output record a log is produced and stored on DBIEXPORT with a sequential key concatenated to the extract id. Click the View Extract Details button to view these log records. They can also be deleted.	B.MAINT
Settings	Click to view default settings.	B.SETTINGS
Extract Id	Click to select from a list of extract definition records from DBIEXPORT.	B.SELECTKEY1
Id	The user-defined code assigned to a particular data extract definition. The definition will be stored under this Id and can be recalled and used multiple times.	DBIEX.ID.WK
Extract Id	The user-defined code assigned to a particular data extract definition. The definition will be stored under this Id and can be recalled and used multiple times.	DBIEX.ID
Show Output	Click to toggle the view from the grid defining the fields to extract back to the sample display of the output record.	B.DISPLAY
Show Grid	Click to toggle the view from the sample display of the output record back to the grid defining the fields to extract.	B.GRID
Run Extract	Click to process the extraction of the data as defined by the file and field names in the grid. The Run Extract button will write the extracted details to the record id named in the Output Record Id field. If the record already exists it will be overwritten. It will also create a log record on DBIEXPORT that can be viewed by clicking the View Extract Details button.	B.EXTRACT
Submit	Click to save the data extraction defintion. The definition can be recaled and run again or modified and run again.	B.PRE.SUBMIT
Delete	Deletes the currently displayed record from the database.	B.DELETE
Clear	Clears all fields currently displayed on the form.	B.CLEAR
Submit	Submits the currently displayed record. The database is updated.	B.SUBMIT
Description	A description of the data extract definition.	DBIEX.DESC
Progress	Display phantom progress.	DBIEX.PH.PROGRESS.WK
Progress Message	Progress message to keep user abreast of the phantom's status.	DBIEX.PH.PROGRESS.MSG.WK
Source File	The name of the primary source file from which fields will be extracted. Fields can also be extracted from any other file where records can be linked to the source file using field values contained within the records on the source file. See From File below.	DBIEX.SOURCE.FILE
Source File	The name of the primary source file from which fields will be extracted. Fields can also be extracted from any other file where records can be linked to the source file using field values contained within the records on the source file. See From File below.	DBIEX.SOURCE.FILE.WK
Date Created	The date that this record was created.	DBIEX.DATE.CREATED
Created By	The user id of the user who created this record.	DBIEX.CREATED.BY

Output File Name	The name of the file or folder to which the flat file will be written.	DBIEX.OUT.FILE
Output File Name	The name of the file or folder to which the flat file will be written.	DBIEX.OUT.FILE.WK
Date Updated	The date that this record was last updated.	DBIEX.DATE.UPDATED
Output Record Id	The name of the record in the Output File to which the flat file will be written. The following string conversions are applied to the value entered in this field: DATE() or date() is converted to todays date internal format TIME() or time() is converted to current time internal format YEAR() or year() is converted to current year MONTH() or month() is converted to current month YMD() or ymd() is converted to year-month-day (2022-03-04)	DBIEX.OUT.ID
Output Record Id	The name of the record in the Output File that is being viewed.	DBIEX.OUT.ID.WK
Most Recent Output File Line Count	Count of lines (rows) in the file holding the extracted data for the most recent run.	DBIEX.DISP.CNT
Extracted Record Cnt	The number of records extracted and written to the export file.	DBIEX.DISP.CNT.WK
Updated By	The user id of the user who last updated this record.	DBIEX.UPDATED.BY
File Type	The current options are tab separated fields, comma separated fields or PDF. For Tab Separated the output file will have a tab character between each field (column). For Comma Separated the output file will have a comma separating fields and double quotes around each field. Select the Not Specified option if you want a comma separating each field with no double quotes.	DBIEX.FILE.TYPE
Lines to Display	Select whether lines from the start, the middle or the end of the output file are to be viewed for verification.	DBIEX.DISP.FL
How Many Lines	Select the number of lines of the extracted data to display in an on form report to allow verification of the run.	DBIEX.DISP.SAMPLE
Selection Sentence	Type an Access selection statement to be invoked to select the records to be extracted from the source file. If no statement is entered then the default action is to SSELECT the entire source file. Note that several select statements can be entered, separated by a semicolon. Only the first statement, up to the semicolon, will be invoked. It is therefore possible to save several statements for reference, and to manipulate the text for any given extract such that the required select occurs first.	DBIEX.SEL.CMD
Selection Result	(No defined field help)	B.RESULT
Result	The number of records selected by the selection command. This is useful, when creating a new selection command, for checking that the command is correct.	DBIEX.SEL.CMD.RESULT
⌄	Click to move a field to be extracted down the list to change the sequence of extracted data columns.	B.MOVE.DN
⌃	Click to move a field to be extracted up the list to change the sequence of extracted data columns.	B.MOVE.UP
From File	Fields can also be extracted from any other file where records can be linked to the source file using field values contained within the records on the source file. Click on the link in the grid column heading to see a list of available files.	DBIEX.FROM.FILE
Field Name	The name of the field on the source file, or on the linked source file. Click on the link in the grid column heading to see a list of available fields for the file selected in this row in the first grid column. You can enter the field name or just the attribute number of the required field if you know the value. A numeric value is assumed to be an attribute number. If any field property on the file has this attribute number then a field property name will be displayed. If more than one field property has this attribute number then the field name that is displayed may not coincide with the field name that you want to use in order to default the correct column heading. In that case you must enter the field name itself.	DBIEX.FIELD

Seq	A count of the grid rows.	DBIEX.SEQ
|~	When re-sequencing grid rows the row that is to be moved up or down is marked with an asterisk.	DBIEX.MOVE.DISP.WK
Move	Click this column on the row containing the field to be moved up or down. Then click the Up or Down button located above this grid column to move the row. Each click will move the row up or down 1 row.	DBIEX.MOVE.WK
Column Heading	The description of the field that will be used as the flat file column heading.	DBIEX.COL.HEADING
Attribute	The attribute reference of the field to be exported.	DBIEX.ATTR
Output Conversion	If extracted fields need output conversion then enter the conversion code here. Typically date fields will require a conversion such as 'D4/'. Money fields will require 'MD2'. This field will default to the required conversion where possible. Developers can add output conversion codes to the grid that displays when the header process is clicked. Use the submit button to exit in this case so that the codes are saved to the record CONVS held on the DBIPMCODE file.	DBIEX.OUTPUT.CONV
Max Length	Enter a value here only if the length of the extracted field is to be truncated to this defined length when it exceeds this length.	DBIEX.MAX.LEN
MV	Multivalue attribute indicator.	DBIEX.MV
MV Action	Select how multi-valued fields are to be processed. If no action is defined then value marks will be converted to pipe '|'.	DBIEX.MV.ACTION
SMV Action	Select how multi-subvalued fields are to be processed. If no action is defined then sub-value marks will be converted to pipe '|'.	DBIEX.SMV.ACTION
Substitute Character for VM	Fields containing value marks (multivalued fields) can be processed in several ways. One way is to substitute the value mark character with the character defined by this field. A set of multi-values in a given field will then be treated as a text string and exported like any other field.	DBIEX.MV.CHAR
Substitute Character for SVM	Fields containing subvalue marks (multi-subvalued fields) can be processed in several ways. One way is to substitute the subvalue mark character with the character defined by this field. A set of multi-subvalues in a given field will then be treated as a text string and exported like any other field. Note that sub-values within multi-values require the multi-values to be treated in a similar way (that is by defining a substitute character different to the character defined here).	DBIEX.SMV.CHAR
Process After Read	Enter the name of a catalogued subroutine to be called after each record is read. The record will be passed as a single argument. The named subroutine will be called in the following manner: CALL @this.subroutine.name(RECORD). The extract will process the content of the record as amended by this subroutine.	DBIEX.PROC.AFTER.READ
Open Files	Displays a list of the files that are opened to allow the extract to access the specified files.	DBIEX.OPEN.FILES.WK
Run Seq	Each extract run generates a sequence number. The details of the extract run are held on file with a key comprising the user-defined code assigned to a particular data extract definition concatenated to this sequnce number, separated by an asterisk.	DBIEX.RUN.SEQ.WK
User Id	The weblogon of the user that runs the instance of the data extraction definition.	DBIEX.RUN.USER.ID.WK
Out File	The name of the Output File to which the flat file will be written.	DBIEX.RUN.OUT.FILE.WK
Out Id	The name of the record in the Output File to which the flat file will be written.	DBIEX.RUN.OUT.ID.WK
Start Date	The date that the data extraction run commenced.	DBIEX.RUN.START.DATE.WK
Start Time	The time that the data extraction run commenced.	DBIEX.RUN.START.TIME.WK
End Date	The date that the data extraction run completed.	DBIEX.RUN.END.DATE.WK
End Time	The time that the data extraction run completed.	DBIEX.RUN.END.TIME.WK

Elapsed Time	The elapsed time of the data extraction run.	DBIEX.RUN.ELAPSED.WK
Line Cnt	The number of rows in the extracted data record.	DBIEX.RUN.LINE.CNT.WK
Orig Cnt	The line count when this file extract was processed. The current line count may differ if the same output record id has been used in subsequent extracts where the number of lines extracted differs.	DBIEX.RUN.LINE.CNT.ORIG.WK
Display All Runs	Click to display all runs for this extract definition.	DBIEX.DISP.ALL
Email Subject	Text to be used as the email Subject	DBIEX.DBMT.SUBJECT
Email Format	The email format for this workflow email template.	DBIEX.DBMT.FORMAT
Name of Sender	The Sender's name.	DBIEX.DBMT.FROM.NAME
Send To	The list of Send To email addresses separated by semicolon. If blank then the email will be sent to the Managed By Group email addresses.	DBIEX.DBMT.TO
Email Address for Replies	The email address for replies.	DBIEX.DBMT.REPLY.TO.ADDRESS
Copy To	The list of Copy To email addresses separated by semicolon.	DBIEX.DBMT.CC
Domain Email Address	The email address of the Domain which is used as the Email From address. This field should only be populated if the email from address is required to be different to the Email From address in Workflow Control.	DBIEX.DBMT.DOMAIN.ADDRESS
Blind Copy To	The list of Blind Copy To email addresses separated by semicolon.	DBIEX.DBMT.BCC
Send	Click to send the extracted data file email.	B.SEND
Convert	"PDF" , "XLS" or blank. PDF and XLS conversion parameters are the same as those that are currently used by DesignBais.	DBIEX.DBMT.ATTACHMENT.CONVERT
Zip	Set to either "True" or "False". "True" will cause the attachment to be zipped.	DBIEX.DBMT.ATTACHMENT.ZIP
PDF Layout	Specifies whether the conversion of the attachment to PDF is it to be in Portrait or Landscape format.	DBIEX.DBMT.ATTACHMENT.PDF.LAYOUT
File Delimiter	The field delimiter character in the file that is to be converted to Excel. For example this could a comma or a tab. If the character can be typed, such as a comma, then enter the character. For tab enter the number 9. This will be converted to char(9). Any numeric value entered here will be converted to its char() equivalent.	DBIEX.DBMT.ATTACHMENT.XLS.DELIMITER
Worksheet Name	The name for the Excel Worksheet to be created during the conversion.	DBIEX.DBMT.ATTACHMENT.XLS.WKS
Refresh	Cleck to refresh	B.REFRESH
Selected Run	The number of the run that is currently displayed.	DBIEX.SELECT.RUN.WK
Report Definition	The on form report of the extracted data.	R.REPORT1

Linking to an external device such a EFTPOS

Use the script called genericEventCall.

For example using a button using jQuery:

- Get the element object
  DBAJAXCMD<-1>='var myElement=$("input[testResult=':"'1']":'"); '
- Set the value
  DBAJAXCMD<-1>='myElement.val("Fred"); '
- Trigger the change event
  DBAJAXCMD<-1>='genericEventCall(myElement.attr("id"),"change"); '

Note:

1. use a custom attribute to identify the target element
2. use myElement.attr("id") to get the element XML id

Both of the above avoid the need to hard code the id.

The input field may be hidden and/or disabled.