

Micro Focus®

# Modernization Workbench™

---

Batch Refresh Process



Copyright © 2010 Micro Focus (IP) Ltd. All rights reserved.

Micro Focus (IP) Ltd. has made every effort to ensure that this book is correct and accurate, but reserves the right to make changes without notice at its sole discretion at any time. The software described in this document is supplied under a license and may be used or copied only in accordance with the terms of such license, and in particular any warranty of fitness of Micro Focus software products for any particular purpose is expressly excluded and in no event will Micro Focus be liable for any consequential loss.

Micro Focus, the Micro Focus Logo, Micro Focus Server, Micro Focus Studio, Net Express, Net Express Academic Edition, Net Express Personal Edition, Server Express, Mainframe Express, Animator, Application Server, AppMaster Builder, APS, Data Express, Enterprise Server, Enterprise View, EnterpriseLink, Object COBOL Developer Suite, Revolve, Revolve Enterprise Edition, SOA Express, Unlocking the Value of Legacy, and XDB are trademarks or registered trademarks of Micro Focus (IP) Limited in the United Kingdom, the United States and other countries.

IBM®, CICS® and RACF® are registered trademarks, and IMS™ is a trademark, of International Business Machines Corporation.

Copyrights for third party software used in the product:

- The YGrep Search Engine is Copyright (c) 1992-2004 Yves Roumazielles
- Apache web site (<http://www.microfocus.com/docs/links.asp?mfe=apache>)
- Eclipse (<http://www.microfocus.com/docs/links.asp?nx=eclp>)
- Cyrus SASL license
- Open LDAP license

All other trademarks are the property of their respective owners.

No part of this publication, with the exception of the software product user documentation contained on a CD-ROM, may be copied, photocopied, reproduced, transmitted, transcribed, or reduced to any electronic medium or machine-readable form without prior written consent of Micro Focus (IP) Ltd. Contact your Micro Focus representative if you require access to the modified Apache Software Foundation source files.

Licensees may duplicate the software product user documentation contained on a CD-ROM, but only to the extent necessary to support the users authorized access to the software under the license agreement. Any reproduction of the documentation, regardless of whether the documentation is reproduced in whole or in part, must be accompanied by this copyright statement in its entirety, without modification.

U.S. GOVERNMENT RESTRICTED RIGHTS. It is acknowledged that the Software and the Documentation were developed at private expense, that no part is in the public domain, and that the Software and Documentation are Commercial Computer Software provided with RESTRICTED RIGHTS under Federal Acquisition Regulations and agency supplements to them. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of The Rights in Technical Data and Computer Software clause at DFAR 252.227-7013 et. seq. or subparagraphs (c)(1) and (2) of the Commercial Computer Software Restricted Rights at FAR 52.227-19, as applicable. Contractor is Micro Focus (IP) Ltd, 9420 Key West Avenue, Rockville, Maryland 20850. Rights are reserved under copyright laws of the United States with respect to unpublished portions of the Software.

---

# Contents

<b>Chapter: 1</b>	<b>Using the Batch Refresh Process . . . . .</b>	<b>1</b>
	Understanding the Batch Refresh Process . . . . .	1
	Configuring the Batch Refresh Process . . . . .	2
	Preparing Files for Batch Refresh Processing . . . . .	8
	Enabling Parallel Verification . . . . .	8
	Executing the Batch Refresh Process . . . . .	9
	Producing Utilities for BRP . . . . .	12
	Guidelines for BRP Utilities . . . . .	13
<b>Chapter: 2</b>	<b>Using Batch Scripts . . . . .</b>	<b>23</b>
	AddNew.bj . . . . .	24
	AffectedCodeReport.bj . . . . .	25
	AnalyzeProgram.bj . . . . .	26
	ApplyPCF.bj . . . . .	28
	BusinessRulesReport.bj . . . . .	29
	BusinessRulesValidation.bj . . . . .	30
	CheckQueue.bj . . . . .	31
	ClipperDetails.bj . . . . .	32
	ClipperMetrics.bj . . . . .	34
	ClipperMultiSearch.bj . . . . .	35
	ClipperSearch.bj . . . . .	36
	ComplexityReport.bj . . . . .	38
	CreatePCF.bj . . . . .	39
	CreateWS.bj . . . . .	40
	CRUDReport.bj . . . . .	41
	DBA.Cobol.bj . . . . .	42

---

DCE.bj . . . . .	44
DiagramBAV.bj . . . . .	45
DiagramCallie.bj . . . . .	47
DiagramFlowchart.bj . . . . .	48
DiagramTS.bj . . . . .	50
EffortReport.bj . . . . .	52
ExecutiveReport.bj . . . . .	53
ExportDescriptions.bj . . . . .	54
ExportRules.bj . . . . .	55
ExportScreens.bj . . . . .	56
GenCopybooks.bj . . . . .	57
GenScreens.bj . . . . .	59
ImpactReport.bj . . . . .	60
ImpactReportFromList.bj . . . . .	61
ImpExBAV.bj . . . . .	63
ImportRules.bj . . . . .	64
IMS Analysis.bj . . . . .	65
Invalidate.bj . . . . .	68
InventoryReport.bj . . . . .	69
Populate.bj . . . . .	70
ReferenceReport.bj . . . . .	71
Refresh.bj . . . . .	73
Register.bj . . . . .	75
Related.bj . . . . .	76
ResolveDecisions.bj . . . . .	79
RestoreDecisions.bj . . . . .	81
RXP.bj . . . . .	82
SaveDecisions.bj . . . . .	83
SetProject.bj . . . . .	84
Unregister.bj . . . . .	85
UpdateOnly.bj . . . . .	87
Upgrade.bj . . . . .	89
Verify.bj . . . . .	90
Executing Batch Scripts . . . . .	93



---

---

# 1

# Using the Batch Refresh Process

The Modernization Workbench (MW) Batch Refresh Process (BRP) lets you register and verify source files in batch mode. You typically use this process when sources on the mainframe have changed, and you need to synchronize the modified sources with the sources you are working with in MW. You can also use BRP to perform analysis and reporting functions.

## Understanding the Batch Refresh Process

Provides an overview of the Batch Refresh Process.

The Batch Refresh Process (BRP) is a utility that supports the synchronization of sources from a mainframe or enterprise server with the MW repositories representing those sources. BRP is installed with the MW server.

BRP is responsible for updating the workspace with sources provided to it from the mainframe and verifying all unverified sources. Optionally, BRP can be configured to run any required source code pre-processing, as well as certain analysis and reporting functions.

When sources are updated to a workspace, the workbench determines whether or not to load the file. When the name of the incoming file matches the name of a file currently in the workspace, the two files are compared. If they are different,

the incoming file will replace the existing file. If they are the same, no change is made. If the incoming file does not currently have a match in the workspace, the file is added to the default project. The default project is a project with the same name as the workspace. If this project does not exist, it is automatically created.

Updating a source in a workspace causes that source to be invalidated. Any sources that are dependent upon the updated file will also be invalidated. For example, an update to a copybook will cause all the COBOL files that use the copybook to become invalidated. That, in turn, will cause all JCL files that execute the programs in the source files to become invalidated. Once the update phase is completed, all invalidated and unverified sources in the workspace will be verified.

## Configuring the Batch Refresh Process

Describes how to configure the Batch Refresh Process.

The Batch Refresh Process is installed with the MW server. For each workspace it processes, BRP refers to an initialization file containing configuration settings. Use the BRP Configurator in the workbench Administration tool on the workbench server to modify the initialization file settings.

---

### TASK

1. In the MW Administration tool, choose **Administer > Configure BRP**. The BRP Configurator opens.
2. In the Current BRP Configurations pane, choose the BRP configuration file you want to edit and click **Edit**.

*NOTE: If the BRP configuration you want to edit is not listed in the pane, click **Find** to locate the file in the file system.*

3. To create a new configuration, click **Add**. A Select Workspace dialog box opens, where you can specify the workspace (.rwp) file you want to configure for BRP.

*NOTE: To copy a configuration, select it and click **Copy**. To delete a configuration, select it and click **Delete**.*



4. The main BRP Configurator window opens, with a tab for each configuration task:
  - On the General tab, set basic BRP initialization values (required).
  - On the User Exit tab, identify any user exits you have created to extend or modify BRP functionality (optional).
  - On the Advanced tab, enable BRP support for IMS Analysis, Executive Report, and WebGen (optional).
5. When you are satisfied with your entries on each tab, click **OK**.

## Configuring General Settings

Describes how to configure BRP settings on the General tab.

Set required BRP initialization values on the General tab. The settings are described in the table below.

Setting	Description
BRP Run Type	Must be set to "Master".
RMW Install Path	Specifies the path of the MW installation folder.
BRP Install Path	Specifies the path of the BRP installation folder. This folder must contain the Reports, PreparedSources, Staging, Utility, and Working folders. Use override parameters for folders in a different location.
Workspace Path	Specifies the path of the folder for the workspace to be refreshed. This folder is at the same level as the workspace (.rwp) file.
Site	Specifies the site name for this BRP install. This value is written out to the main BRP log and is used for documentation purposes only.
Obsolete Processing	Check this box to turn on obsolete processing. Obsolete processing automatically determines which source files are no longer part of a "live" application and moves them to another project. Sources are determined to be obsolete by virtue of being absent from the set of incoming sources for a BRP run.
Obsolete Project	If <b>Obsolete Processing</b> is selected, specifies the project to which obsolete source files will be moved.

Obsolete Exclusions File	If <b>Obsolete Processing</b> is selected, specifies a text file that lists files that should be ignored during obsolete processing. This mechanism is intended to avoid having MW generated or provided files classified as obsolete. For example, MW-provided system copybooks or DASDL generated copybooks. The text file should be formatted with a single file name per line. This mechanism is also useful when there are sources that are particularly difficult to provide on an ongoing basis or if a source is generated during runtime in the application.
--------------------------	---

## Configuring User Exits

Introduces BRP Configurator settings on the User Exits tab.

Identify user exits you have created to extend or modify BRP functionality on the User Exit tab. A user exit is a point in the standard BRP processing when a user-supplied set of commands is executed. Typically the commands execute utilities that accomplish tasks ranging from source code pre-processing to specialized report generation.

### Understanding Exits

Provides an overview of BRP exits.

There are seven user exits in BRP. Each is named and corresponds to a major division of processing, or step, in a BRP run. The names are listed below in the order they are executed:

- Setup
- Init (Initialization)
- Staging
- Update
- Verification
- Generation
- End

With the exception of the Setup and End user exits, each is executed as the very first task of the corresponding BRP step. For example, in the Generation step the Generation user exit is executed followed by executive report generation and WebGen generation.

There are some essential tasks that occur during the Setup step that make it impractical for user exit execution to be first. The Setup step is where the main BRP log is opened and all parameter values are generated, if necessary, and checked for validity. The Setup user exit occurs after the log file is created, but before parameter values are generated and checked.

The End user exit occurs at the very end of the BRP run. There are no tasks that occur after it other than closing the main BRP log file.

Which user exit should be used to execute a particular piece of functionality depends upon the task that needs to be accomplished. For example, source code pre-processing usually needs to occur prior to the sources being loaded into the workspace. This would make the Staging or Update user exits ideal. However, it is best to do source code pre-processing once all sources are in a single spot and are guaranteed to have proper file extensions. That would eliminate the Staging user exit, since it is during Staging that file extensions are added, if necessary. Therefore, the best place to execute source code pre-processing utilities is the Update user exit.

Other common uses of user exits are to run specific reporting or analysis functions. These typically require that the verification step has been completed. Therefore, the Generation user exit will typically work best for these situations.

## Configuring Exits

Describes how to configure BRP Configurator settings on the User Exits tab.

Configuring a user exit involves two separate tasks:

- Creating a BRP-enabled utility to accomplish the task at hand.
- Pointing the user exit to that utility.

Creating a BRP-enabled utility is a non-trivial task. Guidelines and information on this subject can be found in the section *Producing Utilities for BRP*. Use the User Exit tab of the BRP Configurator to point a user exit to the corresponding BRP-enabled utility.

*NOTE: In the default configuration the Generation user exit is configured and provides a useful example.*

BRP contains anchor points for all seven user exits. The DOS batch file should be named for the user exit it corresponds to and it should be located in the BRP Utilities folder. Be sure to specify a full and complete path to the DOS file. Relative paths may not work properly in this context.

The DOS batch file must contain the actual commands that the user exit will execute. This also provides the opportunity to do more than one task in any given user exit.

BRP checks any enabled user exit INI file parameter value for validity during the Setup step. If the value does not point to an existing file, BRP will quit with a severe error.

## Configuring Advanced Settings

Describes how to configure BRP Configurator settings on the Advanced tab.

Configure settings on the Advanced tab to improve verification performance and enable support for IMS Analysis, Executive Report, and WebGen. The settings are described in the table below.

Setting	Description
Launch standalone HyperCode Converter	Check this box to launch the HyperCode Converter. Using the HyperCode Converter generally improves verification performance.
Number of extra HyperCode Converters	If <b>Launch standalone HyperCode Converter</b> is selected, click the arrow buttons to specify the number of additional HyperCode Converters you want to launch.
Wait HyperCode Converter queue	Check this box to force BRP to wait until the HyperCode Converter(s) queue is empty.
Timeout in minutes	If <b>Wait HyperCode Converter queue</b> is selected, click the arrow buttons to specify the time in minutes BRP should wait for the count on the HyperCode Converter(s) queue to change. If the count does not change within the specified time, BRP resumes. Sixty minutes is recommended.
Drop and restore indices	<p>Dropping repository indexes generally improves verification performance when a large number of files need to be verified. Select:</p> <ul style="list-style-type: none"> <li>• <b>Auto</b> if you want BRP to drop repository indexes based on the number of files that need to be verified.</li> <li>• <b>Yes</b> if you want BRP to drop repository indexes.</li> <li>• <b>No</b> if you do not want BRP to drop repository indexes.</li> <li>• Dropped indexes are restored when the verification process completes.</li> </ul>
Run IMS Analysis	Check this box to enable IMS Analysis.
Run WebGen	Check this box to enable WebGen.
Run Executive Report	Check this box to enable Executive Report.

Setting	Description
Report Folder	Specifies the folder to store the Executive Report in. Use the Browse button to locate the folder.
Debug	Check this box to enable debug mode, in which additional messages are written to the BRP log.

## BRP Logging

Describes BRP logging.

At the beginning of every BRP run a timestamp value is generated, consisting of the date and time. That timestamp is used throughout the run in order to uniquely identify and group the logs and information generated. Each run creates log files as well as other pieces of information. The timestamp for the run is added to the beginning of the filename for each log. All logs and information files are typically written to the Reports folder of the BRP install directory.

The main BRP log is, by default, named BRP log.txt, although the name can be altered by changing the command that is found in the runBRP.bat file in the BRP install directory. If more than one BRP installation is present, add the name of the workspace being refreshed to the main BRP log name.

The main BRP log contains basic information on when each major step of the run starts and finishes, as well as any relevant summary or diagnostic information. This is the log to check to determine whether the BRP run completed successfully or not. A run to completion will result in the last message in the log indicating the process finished successfully. Log messages marked "ERROR" should be reviewed. These are problems that were encountered, but they are not bad enough to cause BRP to abend. Messages marked as "SEVERE" are issues encountered that required BRP to abend. These should be investigated and corrected.

In addition to the main BRP log, the Update Log.txt and Verify Log.txt are also generated. These, as their names indicate, document the results of the update and verification steps respectively. The update log contains an entry for each file that is added to the workspace, whether by virtue of being different (updated) or new (added). Files that are processed during update that have unknown file extensions will also be documented in this log. Files that are processed and rejected because they are not different from the version in the workspace are not documented. The verification log lists the verification status of each file that is processed during verification. Summary statistics appear at the end of the log.

The remainder of the logs and files that are generated during a BRP run are there to provide in-depth information for troubleshooting if there is a problem during the run. If there is a problem that requires the attention of support

services, please be sure to include all the logs and files from a run. Sorting the file names in the Reports directory by name will naturally group them together.

## Preparing Files for Batch Refresh Processing

Describes how to prepare source files for BRP.

The incoming sources must be placed in the BRP PreparedSources folder with appropriate file extensions.

If files do not have appropriate file extensions, they must be separated, by type, into individual folders in the PreparedSources folder. Each subfolder must be named for the type of source it contains. Source names must match those used in the Sources folder in the workspace directory. For example, if Cobol sources do not have file extensions they must be placed in PreparedSources\Cobol.

## Enabling Parallel Verification

Describes how to enable parallel verification.

Parallel verification typically improves verification performance for very large workspaces by using multiple execution agents, called *HyperCode Converters*, to process source files concurrently. You can start any number of converters on the local machine, remote machines, or some combination of local and remote machines. You can run parallel verification online in the Modernization Workbench or in batch mode with the Batch Refresh Process (BRP).

You enable parallel verification in three steps:

- Select the parallel verification method and the minimum number of concurrent converters on the Verification > Parallel Verification tab of the Workspace Options.
- Start the converters on the local and/or remote machines. If you start fewer than the minimum number of converters specified on the Parallel Verification tab, the verification process starts the needed converters automatically on the local machine.
- Verify the workspace online in the Modernization Workbench or in batch mode using the Batch Refresh Process (BRP).


*NOTE: Verification results are reported in the Activity Log History window. They are not reported in the Activity Log itself (for online verification) or BRP log files (for batch verification). You can also use a Verification Report to view the results.*

Follow the instructions below to launch HyperCode Converters and to specify the type of work the converters perform. You can launch multiple converters on the same machine. Once the minimum number of converters has been started, you can launch the converters at any point in the verification process.

---

**TASK**

1. In the Modernization Workbench Administration window, choose **Administer > Launch HyperCode Converter**. The Launch HyperCode Converter window opens.
2. In the **Serve workspace** combo box, specify the workspace to be processed.
3. In the Processing Mode pane, select any combination of:
  - **Conversion** to perform operations used to generate a HyperView construct model.
  - **Verification** to perform verification operations.
4. Select **Produce Log File** to generate a log file for parallel verification. The log file has a name of the form `<workspace_name>HCC.<random_number>.log` and is stored at the same level as the workspace (.rwp) file.
5. Click **OK**.

*STEP RESULT:* The workbench launches the HyperCode Converter. Click the  button on the Windows toolbar to view the HyperCode Converter window.

*NOTE: Once verification has started, you can change the processing mode for a converter by selecting the appropriate choice in the **Processing** menu in the HyperCode Converter window.*

---

## Executing the Batch Refresh Process

Describes how to execute BRP.

The BRP Configurator creates a runBRP.bat file and saves it to the location specified in the BRPInstallPath configuration option. Executing this batch file will start a BRP run.

The batch file executes the runBRP.exe executable file with appropriate parameters. The command format is as follows:

```
runBRP.exe <INI file> <log file>
```

where *INI file* is the path to the BRP initialization file and *log file* is a path to the main BRP log file.

**NOTE:** *The workspace is locked while BRP runs. It cannot be accessed by users. In the event of BRP failure, you can unlock the workspace by choosing **Administer > Unlock Workspace** in the Administration tool.*

A full BRP run will produce several detailed log files in addition to the main BRP log. These detail files will always be written to the Reports folder. The main BRP log also is written to the Reports folder by default.

**NOTE:** *Running multiple BRP processes simultaneously on the same workspace is not supported.*

## Adding Source File Extensions

Describes how to BRP add source file extensions.

It is recommended that source files coming into BRP have proper file extensions already in place. In some cases, however, this is not possible and BRP can add them if needed. There is no need to configure initialization file parameters to use the functionality.

To have BRP add the file extensions, you must separate the sources, by type, into separate folders in the PreparedSources directory. Each folder must be named for the source type it contains and the source type name must correspond to MW source type names.

MW source type names can be determined by examining the folder names found in the Sources folder of a workspace directory. If the workspace already contains a source of a particular type, there will be a folder in the Sources directory corresponding to that source type. For example, Cobol files are found in the Cobol folder. The precise file extension that is added for any particular source type is determined by the configuration of the Registration Extensions tab in the target workspace's workspace options. The first defined file extension for each source type will be the extension that is added by BRP. For example, Cobol File has three default file extensions listed: .cbl, .cob, and .ccp. Since .cbl is listed first, that is the extension used by BRP. The order that these values appear in the workspace options can be changed by removing extensions and adding them back in.

Note that file extensions are added onto the file without regard for any currently existing file extension if this functionality is used. For example, if the files in a folder named Cobol currently have a .txt extension (which is commonly added



by some mainframe FTP applications), each file would end up having an extension like .txt.cbl. Various source file naming conventions include multiple "dots" in the source name. Since this scenario is unpredictable and varies widely, it is risky and impractical to have BRP strip any possible existing file extensions.

If there is a mix of sources with and without file extensions, BRP can handle this. Any files with proper extensions should be placed in the PreparedSources directory directly, as normal. Any files that need extensions should be dealt with as described above.

## BRP Logging

Describes BRP logging.

At the beginning of every BRP run a timestamp value is generated, consisting of the date and time. That timestamp is used throughout the run in order to uniquely identify and group the logs and information generated. Each run creates log files as well as other pieces of information. The timestamp for the run is added to the beginning of the filename for each log. All logs and information files are typically written to the Reports folder of the BRP install directory.

The main BRP log is, by default, named BRP log.txt, although the name can be altered by changing the command that is found in the runBRP.bat file in the BRP install directory. If more than one BRP installation is present, add the name of the workspace being refreshed to the main BRP log name.

The main BRP log contains basic information on when each major step of the run starts and finishes, as well as any relevant summary or diagnostic information. This is the log to check to determine whether the BRP run completed successfully or not. A run to completion will result in the last message in the log indicating the process finished successfully. Log messages marked "ERROR" should be reviewed. These are problems that were encountered, but they are not bad enough to cause BRP to abend. Messages marked as "SEVERE" are issues encountered that required BRP to abend. These should be investigated and corrected.

In addition to the main BRP log, the Update Log.txt and Verify Log.txt are also generated. These, as their names indicate, document the results of the update and verification steps respectively. The update log contains an entry for each file that is added to the workspace, whether by virtue of being different (updated) or new (added). Files that are processed during update that have unknown file extensions will also be documented in this log. Files that are processed and rejected because they are not different from the version in the workspace are not documented. The verification log lists the verification status of each file that is processed during verification. Summary statistics appear at the end of the log.

The remainder of the logs and files that are generated during a BRP run are there to provide in-depth information for troubleshooting if there is a problem during the run. If there is a problem that requires the attention of support services, please be sure to include all the logs and files from a run. Sorting the file names in the Reports directory by name will naturally group them together.

## Producing Utilities for BRP

Introduces BRP utilities.

The following are guidelines for producing utilities for BRP. These guidelines apply for any utility. Currently these utilities are normally written by support services and partners.

### Versioning

Describes how to assign a version to a BRP utility.

Each utility needs a version number. The version number should be the date of the last modification made to the utility, formatted as follows:

yyyymmdd

The version number must appear in the first line of the log file that the utility produces.

### Logging

Describes logging for BRP utilities.

Log files are often the only way to get reliable data. The task of analyzing output can become easier when the log files are used and recording appropriate levels of output.

At a minimum log files need to contain:

- Utility name
- Utility version
- Parameter names and values
- Record of files modified/written (when appropriate)

- Record of individual changes made to modified files (when appropriate)

The log message format should be as follows:

```
hh:mm:ss<tab>message type<tab>message
```

Message types can include INFO, WARN, ERROR, SEVERE or DEBUG. These are generally self-explanatory, but SEVERE should not be used unless there is an abend (in Perl the die() command). Add new message types if the situation calls for it. For example, BRP has a SETUP message type.

## Source, Executable, and CFG Files

Describes how utilities are delivered.

Utilities are produced by support services and partners and are delivered as a compiled executable with documentation and, if necessary, a CFG file.

## Guidelines for BRP Utilities

Introduces utilities that need to be enabled for BRP.

This section focuses on guidelines for utilities that need to be "enabled" for BRP. The only difference is where input is coming from, output is going to, and how parameters are provided.

## BRP and Non-BRP Modes

Explains BRP and non-BRP modes.

In general, any utility created for BRP should also be able to be run in a stand-alone manner; that is, it should run outside of and separate from BRP as well. Typically this means getting parameters from a CFG file. This is already being done for all pre-processing type utilities right now. There are occasional situations where this is not practical. The utility needs to be able to determine whether it is being executed in a BRP context or not. If the stand-alone mode requires a CFG file, the absence of a CFG file parameter can serve as a trigger for BRP-mode execution. Where this will not work, the first parameter of the utility should be "BRP" to trigger BRP-mode execution.

## Using User Exits

Describes exit usage

There are several user exit points in BRP. At different user exits potential input files are in different places and output requirements are different as well. Knowing which user exit a utility is going to be run from is crucial. It is recommended that support services be consulted regarding which user exit to employ for a particular task. The majority of user exit utilities are source code pre-processors and all use the Update user exit.

## Parameter Data

Describes parameter data for BRP utilities.

Parameter data typically comes from any of three general sources: command line, CFG file, or DOS environment variables. The first two are straightforward. DOS environment variables are easily acquired by capturing the output of the DOS set command with the following line of Perl code:

```
$dos_env_vars_str = `set`;
```

*NOTE: The special characters preceding and following the word "set" are not single quote characters; they are "backtick" characters.*

The parameter values that drive a BRP run are made available to a user exit via DOS environment variables. BRP generates a DOS batch file that contains commands to set DOS environment variables. The user exit command is added to the end of the generated batch file and the batch file is executed using the backtick operator in Perl. The backtick operator executes a DOS command (in BRP the path to a batch file) in a shell "nested" inside of the shell of the BRP executable. The environment variables set up for a user exit only exist during the execution of that user exit. The environment variable commands are re-generated and run for each user exit.

In general, the format of parameter names and values should be standardized. BRP job parameters are of the form:

*Parameter Name = Parameter Value*

DOS environment variables and CFG file parameters are formatted in the same way. Command line parameters should follow the same standard. In general, command line parameters need to override the same named parameter from a CFG file or DOS environment. This allows a way to alter behavior in cases where the user may not have direct control over all the values.

## Logging

Describes logging for BRP-mode utilities.

In addition to the general logging guidelines, the name of the log file and where it is written need to be addressed in BRP-mode utilities.

The log file name pattern is:

```
(timestamp)UtilityName Log.txt
```

where *UtilityName* is obvious and *timestamp* is a BRP environment parameter (BRP\_TIMESTAMP) that identifies all logs for a BRP run.

## Input/Output

Describes input/output for BRP utilities.

Input and output locations will change depending on what files are needed and which user exit the utility is run from. Most utilities (source code pre-processing) will be running from the Update user exit.

The sources coming into the BRP process will be in the following path:

```
BRP_STAGINGDIR\BRP_TIMESTAMP
```

where *BRP\_STAGINGDIR* is a full path referring to the Staging folder of a BRP install and *BRP\_TIMESTAMP* holds the timestamp value for the current BRP run.

Output sources must be written back to this same location. However, to maintain integrity should the user-exit utility fail or otherwise not finish, it is recommended that output sources be written to the BRP Working folder (*BRP\_WORKINGDIR*) and only when processing is completed should they then be copied back to the proper output location. A subfolder should be created in the Working folder for this purpose using the following format:

```
timestamp_UtilityName
```

This naming convention is required.

## Returning Values

Describes returning values for BRP utilities.

BRP determines the return state of a user exit by examining all the output written to the "console" (STDOUT in Perl terminology) by the commands executed by the user exit. The examination is done after the user exit completes

execution and control returns back to the BRP run. If there is no output BRP assumes the user exit commands completed successfully. If there is any output found BRP assumes there was a SEVERE level error and will immediately stop the run.

BRP will include any output it finds in a SEVERE level message in the main BRP log. Any user exit executed utility should be sure to make effective use of this behavior. User exits do not have any knowledge of what commands or utilities they are executing. Therefore a message written to the console should contain the utility or command name along with an appropriately brief message. The details behind a utility failure can be included in the utility's own log.

## BRP Environment Parameters

Describes BRP environment parameters.

BRP parameter values are split into two groups. Ones prefixed with "BRP\_" are for BRP specific values. Those prefixed with "EXT\_" are for source file extension definitions. The table below lists all variables that are set by BRP for use by user exits along with a short description. Any path value will be fully qualified unless otherwise noted.

Category	Name	Description
Timestamp	BRP_TIMESTAMP	Timestamp value that uniquely identifies a BRP run and the logs that are generated during that run.
BRP Logs		These parameters are the paths to log files from running various BRP. Note that all the references to specific jobs are default settings only. There are very few cases where these exact jobs will not be used, but they do exist.
	BRP_APPLYOBSOLETEPCFLOGFILE	Log from <i>ApplyPCF.bj</i> job for applying the <i>BRP_OBSOLETEPCF</i> file.
	BRP_BWGLOGFILE	Log from <i>BWG.exe</i> (Batch WebGen).
	BRP_CREATEBEGINPCFLOGFILE	Log from <i>CreatePCF.bj</i> job. This is run at the beginning of the BRP run and creates the file <i>BRP_BEGINPCF</i> .
	BRP_CREATEENDPCFLOGFILE	Log from <i>CreatePCF.bj</i> job. This is run just before verification and creates the file <i>BRP_ENDPCF</i> .

Category	Name	Description
	BRP_EXECREPORTLOGFILE	Log for the ExecutiveReport.bj job.
	BRP_GETEXTLOGFILE	Log for the GetExtensions2.mbu job. This is run at the beginning of the BRP and creates the file BRP_FILEEXTFILE. See the section below on extension values for more information.
	BRP_IMSANALYSISLOGFILE	Log for the IMS Analysis.bj job.
	BRP_UPDATELOGFILE	Log for the UpdateOnly.bj job.
	BRP_VERIFYLOGFILE	Log for the VerifyOnly2.bj job.
PCF files		These are parameters for the various PCF files that are generated and used during a BRP run.
	BRP_BEGINPCF	Generated at the beginning of a BRP run. Used for many purposes in BRP including determining obsolete sources.
	BRP_ENDPCF	Generated toward the end of a BRP run, after the BRP_OBSOLETEPCF is applied and before verification.
	BRP_OBSOLETEPCF	Generated during a Master BRP run if BRP_OBSOLETEPROCESSING is set to 1. This will shift sources missing from the current incoming set of files to the project specified in BRP_OBSOLETEPROJECT.
BRP Files, Folders, Flags, and Log		These parameters are data files that BRP uses, flags that turn certain processing on or off, BRP install folders and other various values.
	BRP_DROPIND	Flag (1/0) that drops database indexes to improve verification and IMS Analysis performance.
	BRP_LAUNCHHCC	Flag (1/0) that launches the HyperCode Converter to improve verification performance.
	BRP_WAITHCC	Time in minutes to wait for HyperCode Converter to respond.

<b>Category</b>	<b>Name</b>	<b>Description</b>
	BRP_BRPLOGFILE	Main log file for a BRP run.
	BRP_BRPINSTALLPATH	Path where BRP is installed. This is specified in the BRP initialization file. BRP will derive the path values for the six BRP folders (PreparedSources, Reports, Staging, Utilities and Working) based off this path if they are not specified in the initialization file.
	BRP_PREPAREDSOURCESDIR	Path where sources coming into BRP start off. In cases where a utility is in place to handle getting the sources off a server or mainframe this is the location where those sources are copied to.
	BRP_REPORTDIR	Path to the folder where all log files are written to. Other resource files created during a BRP run are also written here including all PCF files, file extension data file and all generated user exit batch files.
	BRP_RMWINSTALLPATH	Path to the install folder for MW.
	BRP_STAGINGDIR	Folder where sources reside for updating to the workspace. Sources will actually be in a subfolder that is named with the BRP timestamp and not the Staging folder directly. This is also where the majority of source pre-processing utilities will look for inputs and write outputs.
	BRP_UTILITIESDIR	Path to directory that contains all the executables the BRP will need along with extra resource files and the static user exit batch files.
	BRP_WORKINGDIR	Folder user exit utilities should use for any work they need to perform.
	BRP_WORKSPACEDIR	Path to the target workspace.
	BRP_FILEEXTFILE	File containing the file extension definitions for the target workspace.



Category	Name	Description
	BRP_LASRUNFILE	Text file containing the timestamp of the last BRP run that completed execution.
	BRP_OBSOLETEEXCLUSIONSFILE	File that lists any files that should be excluded from obsolete processing. Typically this includes files generated by MW (ex. DASDL copybooks), but is often used for client-specific sources as well.
	BRP_BRPRUNTYPE	Must be set to "Master".
	BRP_DEBUG	Flag (1/0) that will increase the amount of messaging written to the main BRP log. Typically this is always set to 1.
	BRP_LASRUNTIMESTAMP	The timestamp value of the last BRP run that completed execution.
	BRP_OBSOLETEPROCESSING	Flag (1/0) that turns obsolete processing on or off. When it is turned on the incoming set of files will be compared against the set of file currently in the target workspace. Any files currently in the workspace, but not in the incoming set of files will be moved to an obsolete project (named in the BRP_OBSOLETEPROJECT parameter).
	BRP_OBSOLETEPROJECT	Name of a project where obsolete sources will be moved to.
	BRP_SITE	Documentation parameter that is set in the BRP initialization file. The value here will be written to the beginning of the main BRP log file. It is used mainly for support purposes to ensure that initialization and log files produced by a client match up.

Category	Name	Description
User Exits	BRP_USEREXIT_SETUP BRP_USEREXIT_STAGING BRP_USEREXIT_INIT BRP_USEREXIT_UPDATE BRP_USEREXIT_VERIFICATION BRP_USEREXIT_GENERATION BRP_USEREXIT_END	These parameters contain the command that will be executed by BRP. Typically this will specify a static DOS batch file (as opposed to the generated batch file BRP generates for each user exit). The static batch file is used so that multiple commands can be executed in a single user exit. These parameters will only exist if the user exit is being used. Below is a complete list of all user exit parameters, but it will be rare to see them all at once. They are listed in the order they would be executed in a BRP run. There is one user exit for each major step of the BRP. They always are the first task that is done in each step.
File Extensions		These parameters contain information on the source file extensions that are valid for each legacy file type for the target workspace. The exact parameters that will be here depend upon what options are activated for the target workspace.
	Parameter Names	The general format of the parameter names is EXT_ <i>type</i> , where <i>type</i> is the name of the corresponding directory in the workspace Sources folder. Note that in the past this name is not necessarily the same as the type name found in a PCF file. For example, PL/I-included sources are contained in the Sources folder PLIInclude, but that source type is named "PLINC" in PCF files.
	Parameter Values	The extension values are separated by a single space and will be in the same order they appear in the workspace options window. Example: EXT_COBOL = cbl cob ccp C74 .

## Testing

Provides guidance for testing BRP utilities.

Testing user exit utilities can be challenging. The easiest way to do this is to use one of the DOS batch files generated for each enabled user exit during a BRP run. These files will contain all the parameters and the values can be changed to suit the needs of the testing requirements. Replace the last command in the file with whatever command is necessary. In a Perl context we would use:

```
perl -d myUtility.pl
```

Note that a DOS command window will not execute one of these generated batch files when they have same filename the BRP run assigns (for example, (timestamp)UserExit.bat). This is due to some intrinsic interpretation of the leading "(timestamp)" in the file name. Simply delete this portion of the file name and the batch file will work normally.



---

# 2

# Using Batch Scripts

Use the batch job scripts supplied with Modernization Workbench in BRP user exits or on a standalone basis. The scripts are located in \<Workbench Home>\Scripts\BRP.

Only scripts recommended for use by clients are described. Unless otherwise specified, tool options set in the Modernization Workbench govern the behavior of the scripts.

The *notification file* available in some scripts summarizes job execution. A sample notification file for the Verify.bj script follows. The notification file indicates that eight source files were verified successfully and two were verified with errors.

```
Date: 10/28/2009
Workspace C:\Workspaces\Training
Status of the Verification step:
successful - 8
with errors - 2
failed - 0
```

# AddNew.bj

Register new source files only.

## Action

Register new source files only. Use:

- Register.bj to register new source files and refresh updated source files.
- UpdateOnly.bj to refresh updated source files only.
- Refresh.bj to register and verify new and updated source files.
- Verify.bj to verify registered source files.

## Syntax

```
AddNew Workspace Dir [Entity] [Project] [Detailed]
```

### Required Parameters

Workspace

Dir

### Description

Workspace file (.rwp).

Staging directory for incoming source files.

### Optional Parameters

Entity

Project

Detailed

### Description

\* or entity type of source files to register.  
Default is \*.

Project to register source files in.

Log file.

## Example

```
AddNew C:\Workspaces\Training.rwp E:\StagingArea COBOL  
C:\log.txt
```

# AffectedCodeReport.bj

Generate an Affected Code Report.

## Action

Generate an Affected Code Report in MS Word format. The report shows code that would be impacted by changing a data item's definition or usage. The data item is called a *seed field*.

## Syntax

```
AffectedCodeReport Workspace Model SearchPattern CriterionName
[Accumulate] [Detailed]
```

### Required Parameters

	Description
Workspace	Workspace file (.rwp).
Model	HyperView model for the source files to be searched.
SearchPattern	Search criterion for the seed field.
CriterionName	Name of the search criterion for the seed field.
Accumulate	Whether to append the report to existing reports, True or False. Default is False.

### Optional Parameters

	Description
Detailed	Log file.

## Example

```
AffectedCodeReport C:\Workspaces\Training.rwp COBOL "Name LIKE
*CUSTMAS" Custmas C:\log.txt
```

# AnalyzeProgram.bj

Generate HyperView information.

## Action

Generate HyperView information for the workspace.

## Syntax

```
AnalyzeProgram Workspace [Project] [Notify] [Detailed] [Drop]  
[LaunchHHC] [ExtraHHC] [StopHHC] [Wait]
```

### Required Parameters

Workspace

### Description

Workspace file (.rwp).

### Optional Parameters

Project

### Description

Project to generate HyperView information for.

Notify

Notification file.

Detailed

Log file.



### Oracle Only Parameters

Drop

### Description

Whether to drop repository indexes.  
Specify:

- Auto, to let the script determine whether to drop repository indexes based on the number of files to be processed.
- Yes, to drop repository indexes.
- No, to not drop repository indexes.

Dropping repository indexes generally improves performance when a large number of files need to be processed. Dropped indexes are restored when processing is complete.

LaunchHHC

Whether to launch the HyperCode Converter, Yes or No. Launching the HyperCode Converter generally improves performance.

ExtraHHC

If LaunchHHC is specified, the number of additional HyperCode Converters to launch.

StopHHC

Whether to stop the HyperCode Converter(s) when processing is complete, Yes or No.

Wait

Whether to wait until the HyperCode Converter(s) queue is empty. Specify:

- Yes, to wait indefinitely.
- No, to not wait.
- The number of seconds to wait for the count on the queue to change. If the count does not change within the specified time, BRP resumes. Sixty minutes is recommended. 0 means no timeout.

### Example

```
AnalyzeProgram C:\Workspaces\Training.rwp C:\log.txt Auto Yes
Yes 3600
```

# ApplyPCF.bj

Assign source files to projects based on a project control file (PCF).

## Action

Assign source files to projects based on a *project control file (PCF)*. A project control file identifies the projects to which source files belong.

ApplyPCF.bj differs from SetProject.bj in that it does not allow you to assign source files to projects additively. Use CreatePCF.bj or Related.bj to create a project control file.

## Syntax

```
ApplyPCF Workspace ProjectCF [Detailed]
```

### Required Parameters

Workspace

ProjectCF

### Description

Workspace file (.rwp).

Project control file (.pcf).

### Optional Parameters

Detailed

### Description

Log file.

## Example

```
ApplyPCF C:\Workspaces\Training.rwp E:\Training.pcf C:\log.txt
```

# BusinessRulesReport.bj

Generate a Business Rules Report.

## Action

Generate a Business Rules Report. The report lists the business functions, rule sets, segments, attributes, data elements, and control conditions of business rules in the workspace.

## Syntax

```
BusinessRulesReport Workspace File [Project] [Detailed]
```

### Required Parameters

Workspace

File

### Description

Workspace file (.rwp).

Output file. The format of the report depends on the extension. Supported extensions are .html, .htm, .xls, .rtf, .doc, .txt, and .csv.

### Optional Parameters

Project

Detailed

### Description

Project to generate the report for.

Log file.

## Example

```
BusinessRulesReport C:\Workspaces\Training.rwp  
C:\BusinessRules.htm C:\log.txt
```

# BusinessRulesValidation.bj

Validate business rules.

## Action

Indicate whether business rule segments no longer are valid. An invalid segment is out of synch with the rule, typically because lines of code have been added or deleted during a refresh or edit. Optionally, specify how to handle invalid segments.

## Syntax

```
BusinessRulesValidation Workspace [Action] [Project] [Detailed]
```

Required Parameters	Description
Workspace	Workspace file (.rwp).

### Optional Parameters

Action

### Description

How to handle invalid segments. Specify:

- leave, to retain the invalid segment but set the rule Segment Validity attribute to Invalid.
- delete, to delete the invalid segment and set the rule Segment Validity attribute to Invalid.
- valid, to resynchronize the segment, subject to the limitations described in *Analyzing Programs* in the workbench documentation set.

The values are case-sensitive.

Project

Project to validate business rules for.

Detailed

Log file.

### Example

```
BusinessRulesValidation C:\Workspaces\Training.rwp leave
C:\log.txt
```

## CheckQueue.bj

Check whether the HyperCode Converter(s) queue is empty.

### Action

Check whether the HyperCode Converter(s) queue is empty.

### Syntax

```
CheckQueue Workspace [Project] [Detailed] [Wait]
```

### Required Parameters

Workspace

### Description

Workspace file (.rwp).

### Optional Parameters

Project

### Description

Project to check queue for.

### Oracle Only Parameters

Wait

### Description

Whether to wait until the HyperCode Converter(s) queue is empty. Specify:

- Yes, to wait indefinitely.
- No, to not wait.
- The number of seconds to wait for the count on the queue to change. If the count does not change within the specified time, BRP resumes. Sixty minutes is recommended. 0 means no timeout.

### Example

```
CheckQueue C:\Workspaces\Training.rwp C:\log.txt 3600
```

# ClipperDetails.bj

Generate a Clipper Details Report.

## Action

Generate a Clipper Details Report. For each source file in the specified Clipper list, the report shows the constructs in the list, their type, and their location in the file. You can customize the report to include any HyperView attribute related to the constructs.

## Syntax

```
ClipperDetails Workspace Model ListName Category FileName Attrs
[Project] [Detailed]
```

Required Parameters	Description
Workspace	Workspace file (.rwp).
Model	HyperView model for the source files in the list.
ListName	Name of the list. The list is assumed to be owned by the current user. If it is owned by another user, append a vertical bar ( ) and the user name.
Category	Category of the list.
FileName	Output file. The format of the report depends on the extension. Supported extensions are .html, .htm, .xls, .rtf, .doc, .txt, and .csv.
Attrs	String containing HyperView attributes to include in the report, separated by a vertical line ( ).
Optional Parameters	Description
Project	Project to generate report for.
Detailed	Log file.

### Example

```
ClipperDetails C:\Workspaces\Training.rwp COBOL Miscellaneous
General C:\ClipperDetails.htm Faked|NestingLevel C:\log.txt
```

# ClipperMetrics.bj

Generate a Clipper Metrics Report.

## Action

Generate a Clipper Metrics Report. For each list in the specified Clipper category, the Metrics Report shows the number of list items in each program in the workspace.

## Syntax

```
ClipperMetrics Workspace Model Category FileName [Project]
[Detailed]
```

### Required Parameters

	Description
Workspace	Workspace file (.rwp).
Model	HyperView model for the source files in the lists.
Category	Category of the list.
FileName	Output file. The format of the report depends on the extension. Supported extensions are .html, .htm, .xls, .rtf, .doc, .txt, and .csv.

### Optional Parameters

	Description
Project	Project to generate report for.
Detailed	Log file.

## Example

```
ClipperMetrics C:\Workspaces\Training.rwp COBOL General
C:\ClipperMetrics.htm C:\log.txt
```



# ClipperMultiSearch.bj

Execute a Clipper search with multiple criteria.

## Action

Execute a Clipper search with multiple criteria. The results are displayed in the specified Clipper list.

## Syntax

```
ClipperMultiSearch Workspace Criteria Model ListName Category
[Project] [Accumulate] [Detailed]
```

Required Parameters	Description
Workspace	Workspace file (.rwp).
Criteria	Full path name of the search criterion in the HyperView Advanced Search tool, including the tab name (General) and any folder names. For example, General > Coding Standards\MOVE Statements\Possible Data Padding. Follow the notation specified exactly.
Model	HyperView model for the source files to be searched.
ListName	Name of the list. The list is assumed to be owned by the current user. If it is owned by another user, append a vertical bar ( ) and the user name.
Category	Category of the list.

Optional Parameters	Description
Project	Project to execute search for.
Accumulate	Whether to append the results to existing results, True or False. Default is False.
Detailed	Log file.

### Example

```
ClipperMultiSearch C:\Workspaces\Training.rwp C:\Program  
Files\Modernization Workbench\Data\CodeDefects.xml COBOL  
Miscellaneous General C:\log.txt
```

# ClipperSearch.bj

Execute a Clipper search.

## Action

Execute a Clipper search. The results are displayed in the specified Clipper list.

## Syntax

```
ClipperSearch Workspace Criterion Model ListName Category  
[Project] [Accumulate] [Detailed]
```

Required Parameters	Description
Workspace	Workspace file (.rwp).

### Required Parameters

	Description
Criterion	Full path name of the search criterion in the HyperView Advanced Search tool, including the tab name (General) and any folder names. For example, General > Coding Standards\MOVE Statements\Possible Data Padding. Follow the notation specified exactly.
Model	HyperView model for the source files to be searched.
ListName	Name of the list. The list is assumed to be owned by the current user. If it is owned by another user, append a vertical bar ( ) and the user name.
Category	Category of the list.

### Optional Parameters

	Description
Project	Project to execute search for.
Accumulate	Whether to append the results to existing results, True or False. Default is False.
Detailed	Log file.

### Example

```
ClipperSearch C:\Workspaces\Training.rwp General:Coding
Standards\MOVE
Statements\Possible Data Padding COBOL Miscellaneous General
C:\log.txt
```

# ComplexityReport.bj

Generate a Complexity Metrics Report.

## Action

Generate a Complexity Metrics Report. The report shows complexity metrics for objects of the specified type.

## Syntax

```
ComplexityReport Workspace File [Entity] [Project] [Detailed]
```

### Required Parameters

Workspace

File

### Description

Workspace file (.rwp).

Output file. The format of the report depends on the extension. Supported extensions are .html, .htm, .xls, .rtf, .doc, .txt, and .csv.

### Optional Parameters

Entity

Project

Detailed

### Description

Entity type of objects to report on. Default is program.

Project to generate the report for.

Log file.

## Example

```
ComplexityReport C:\Workspaces\Training.rwp  
C:\ComplexityMetrics.htm C:\log.txt
```

# CreatePCF.bj

Create a project control file (PCF).

## Action

Create a *project control file (PCF)* for a workspace. A project control file identifies the projects to which source files belong. Use ApplyPCF.bj or SetProject.bj to assign source files to projects based on the project control file.

## Syntax

```
CreatePCF Workspace Out [Detailed]
```

### Required Parameters

Workspace

Out

### Description

Workspace file (.rwp).

Output file (.pcf).

### Optional Parameters

Detailed

### Description

Log file.

## Example

```
CreatePCF C:\Workspaces\Training.rwp E:\Training.pcf C:\log.txt
```

# CreateWS.bj

Create a workspace.

## Action

Create a workspace.

## Syntax

```
CreateWS Workspace DSN Schema Password [User] [TableSpace]  
[IndexSpace] [DB] [Detailed]
```

### Required Parameters

### Description

Workspace

Workspace file (.rwp).

DSN

ODBC data source name (DSN) for the database that holds the repository.

Schema

Database schema name for the repository.

Password

Database password that gives access to the schema.

Optional Parameters	Description
User	Database user name that gives access to the schema.
TableSpace	Name of the tablespace for the repository.
IndexSpace	Name of the tablespace for database indexes.
DB	Database type.
Detailed	Log file.

### Example

```
CreateWS C:\Workspaces\Training.rwp OracleLab lab labuserpwd
labuser C:\log.txt
```

## CRUDReport.bj

Generate a CRUD Report.

### Action

Generate a CRUD Report. The report shows the data operations programs perform (Insert, Read, Update, or Delete) and the data objects on which the programs operate.

### Syntax

```
CRUDReport Workspace File [Project] [Detailed]
```

Required Parameters	Description
Workspace	Workspace file (.rwp).

### Required Parameters

File

### Description

Output file. The format of the report depends on the extension. Supported extensions are .html, .htm, .xls, .rtf, .doc, .txt, and .csv.

### Optional Parameters

Project

### Description

Project to generate the report for.

Detailed

Log file.

### Example

```
CRUDReport C:\Workspaces\Training.rwp C:\CRUD.htm C:\log.txt
```

## DBA.Cobol.bj

Perform Cobol domain-based analysis.

### Action

Perform domain-base analysis of Cobol programs. Domain-based analysis "slices out" a specialized program based on the values of one or more variables.

An input file in CSV format identifies the slice parameters. Each line contains the following information:

```
ProgName, SliceName, DataItem, FileName, Row, Col, Comparison, Value, LowerValue, UpperValue
```

where:

- *ProgName* is the name of the program from which the slice will be extracted.
- *SliceName* is the name of the slice.
- *DataItem* is the name of the specialization variable.



- *FileName* is the name of the source file containing the specialization variable.
- *Row* is the row number of the specialization variable in the source file.
- *Col* is the column number of the specialization variable in the source file.
- *Comparison* is the comparison type: "equals" sets the specialization variable to the values specified in *Value*; "not equals" sets the specialization variable to every value but the values specified in *Value*.
- *Value* is the value to set the specialization variable to.
- If *Value* is omitted, *LowerValue* is the lower value of the range of values to set the specialization variable to.
- If *Value* is omitted, *UpperValue* is the upper value of the range of values to set the specialization variable to.

Multiple locations can be specified for a slice. Multiple conditions can be set for a location. All content is case-sensitive.

### Input File Sample

```
DAYOFWEEK,Domain1,YEAR,DayOfWeek.cbl,12,12,equals,2000,,
DAYOFWEEK,Domain1,YEAR,DayOfWeek.cbl,12,12,equals,,2002,2005
DAYOFWEEK,Domain1,MONTH,DayOfWeek.cbl,13,12,equals,4,,
DAYOFWEEK,Domain1,MONTH,DayOfWeek.cbl,13,12,equals,5,,
DAYOFWEEK,Domain1,MONTH,DayOfWeek.cbl,65,19,equals,5,,
DAYOFWEEK,Domain1,MONTH,DayOfWeek.cbl,65,19,equals,6,,
DAYOFWEEK,Domain1,MONTH,DayOfWeek.cbl,95,15,equals,7,,
DAYOFWEEK,Domain1,MONTH,DayOfWeek.cbl,95,15,equals,,1,3
DAYOFWEEK,Domain2,YEAR,DayOfWeek.cbl,81,15,equals,,2001,2010
GSS,Domain3,GSS1003-CMD-CODE-I,GSS.cbl,186,16,equals,"ENTER",,
```

## Syntax

```
DBA.Cobol Workspace List [Options] [Export] [Notify] [Detailed]
```

Required Parameters	Description
Workspace	Workspace file (.rwp).
List	CSV file with slice parameters.

Optional Parameters	Description
Options	Options script file. Default values for options usually are acceptable. Contact support services for special needs.
Export	Destination folder for slices. Results normally are viewed in MW Component Maker.
Notify	Notification file.
Detailed	Log file.

### Example

```
DBA.Cobol C:\Workspaces\Training.rwp C:\Slices.csv C:\log.txt
```

## DCE.bj

Perform dead-code elimination.

### Action

Perform dead-code elimination (DCE) for programs in source files of the specified type. For each program analyzed for dead code, DCE generates a component that consists of the original source code minus any unreferenced data items or unreachable procedural statements.

### Syntax

```
DCE Workspace Entity [Options] [Pattern] [Project] [Detailed]
```

Required Parameters	Description
Workspace	Workspace file (.rwp).

**Required Parameters**

Entity

**Description**

HyperView model for the source files to be analyzed for dead code. Valid values are COBOL, PLI1, NATURAL, and NATSUBROUTINE.

**Optional Parameters**

Options

**Description**

Options script file. Default values for options usually are acceptable. Contact support services for special needs.

Pattern

Pattern for naming generated components. The pattern may contain any valid symbols. An asterisk (\*) is replaced with the name of the analyzed program. If the argument is omitted, component names are generated in the form BRE $nn$ , where  $nn$  is an incrementing number.

Project

Project to analyze source files in.

Detailed

Log file.

**Example**

```
DCE C:\Workspaces\Training.rwp COBOL *-DCE C:\log.txt
```

# DiagramBAV.bj

Generate Batch Application Viewer (BAV) Diagrams.

## Action

Generate Batch Application Viewer (BAV) Diagrams for the workspace. The diagrams show the relationships between jobs, procedures and programs, and data stores.

## Syntax

DiagramBAV *Workspace* [*Pattern*] [*Project*] [*Detailed*]

### Required Parameters

### Description

Workspace

Workspace file (.rwp).

### Optional Parameters

### Description

Pattern

Pattern for naming generated diagrams, consisting of the output folder, file name pattern, and extension. For example, D:\*.bmp. The file name pattern may contain any valid symbols. An asterisk (\*) is replaced with the name of the analyzed program. The format of the diagrams depends on the extension. Supported extensions are .dgm.xml, .bmp, .jpg, .vsd, .vdx, and .emf. If the argument is omitted, diagram names are generated in the form  
`\WorkspaceFolder\Output\ProgramName.dgm.xml`.

Project

Project to generate the diagrams for.

Detailed

Log file.

### Example

DiagramBAV C:\Workspaces\Training.rwp D:\*.bmp C:\log.txt

# DiagramCallie.bj

Generate Callie Diagrams.

## Action

Generate Callie Diagrams for the workspace. The diagrams show the call flow for paragraphs in a Cobol program, subroutines in an RPG program, or procedures in a PL/I or Natural program.

The *subgraph* mode offers a cyclic representation of the information in the diagram. Items are drawn once. Relationship lines cross. Subgraph views are often easier to understand than subtree views.

The *subtree* mode offers a linear representation of the information in the diagram. Items are drawn as many times as necessary. Relationship lines do not cross. Use this view if too many intersecting lines make a subgraph view hard to read.

## Syntax

```
DiagramCallie Workspace [Pattern] [Mode] [Project] [Detailed]
```

### Required Parameters

### Description

Workspace

Workspace file (.rwp).

### Optional Parameters

### Description

Pattern

Pattern for naming generated diagrams, consisting of the output folder, file name pattern, and extension. For example, D:\* .bmp. The file name pattern may contain any valid symbols. An asterisk (\*) is replaced with the name of the analyzed program. The format of the diagrams depends on the extension. Supported extensions are .dgm.xml, .bmp, .jpg, .vsd, .vdx, and .emf. If the argument is omitted, diagram names are generated in the form `\WorkspaceFolder\Output\ProgramName.dgm.xml`.

Mode

Mode of the diagram, subgraph or subtree.

Project

Project to generate the diagrams for.

Detailed

Log file.

### Example

```
DiagramCallie C:\Workspaces\Training.rwp D:* .bmp subgraph  
C:\log.txt
```

# DiagramFlowchart.bj

Generate Flowchart Diagrams.

## Action

Generate Flowchart Diagrams for the workspace. The diagrams show the flow of control between statements in a Cobol paragraph or PL/I procedure, or between steps in a job or JCL procedure.

## Syntax

DiagramFlowchart *Workspace* [*Pattern*] [*Project*] [*Detailed*]

### Required Parameters

Workspace

### Description

Workspace file (.rwp).

### Optional Parameters

Pattern

### Description

Pattern for naming generated diagrams, consisting of the output folder, file name pattern, and extension. For example, D:\*.\*.bmp. The file name pattern may contain any valid symbols. An asterisk (\*) is replaced with the name of the analyzed program. The format of the diagrams depends on the extension. Supported extensions are .dgm.xml, .bmp, .jpg, .vsd, .vdx, and .emf. If the argument is omitted, diagram names are generated in the form \WorkspaceFolder\Output\ProgramName.dgm.xml.

Project

Project to generate the diagrams for.

Detailed

Log file.

### Example

DiagramFlowchart C:\Workspaces\Training.rwp D:\*.\*.bmp C:\log.txt

# DiagramTS.bj

Generate relationship flow diagrams.

## Action

Generate relationship flow diagrams for the workspace. The diagrams show the relationship flow for every object of the specified type in the specified scope.

## Syntax

```
DiagramTS Workspace Scope [Pattern] [Entity] [Tag] [Layout]  
[Project] [Detailed]
```

### Required Parameters

### Description

Workspace

Workspace file (.rwp).

Scope

Scope of the diagrams (including user-defined scopes).



### Optional Parameters

### Description

Pattern	<p>Pattern for naming the generated diagrams, consisting of the output folder, file name pattern, and extension. For example, D:*.*.emf. The file name pattern may contain any valid symbols. An asterisk (*) is replaced with the name of the analyzed program. The format of the diagrams depends on the extension. Supported extensions are .dgm.xml, .bmp, .jpg, .vsd, .vdx, and .emf. If the argument is omitted, diagram names are generated in the form <code>\WorkspaceFolder\Output\ProgramName.dgm.xml</code>.</p>
Entity	<p>* or entity type of objects to diagram. Default is *.</p>
Tag	<p>Tag used to "black-box" objects in the diagrams.</p>
Layout	<p>Diagram layout: circular, hierarchical, orthogonal, symmetric, or tree.</p>
Project	<p>Project to generate the diagrams for.</p>
Detailed	<p>Log file.</p>

### Example

```
DiagramTS C:\Workspaces\Training.rwp Data Flow D:*.*.emf program
circular C:\log.txt
```

# EffortReport.bj

Generate an Effort Estimation Report.

## Action

Generate an Effort Estimation Report. The report compares source files based on weighted values for selected complexity metrics.

## Syntax

```
EffortReport Workspace File [Project] [Detailed]
```

### Required Parameters

Workspace

File

### Description

Workspace file (.rwp).

Output file. The format of the report depends on the extension. Supported extensions are .html, .htm, .xls, .rtf, .doc, .txt, and .csv.

### Optional Parameters

Project

Detailed

### Description

Project to generate the report for.

Log file.

## Example

```
EffortReport C:\Workspaces\Training.rwp C:\EffortEstimation.htm  
C:\log.txt
```

# ExecutiveReport.bj

Generate an Executive Report.

## Action

Generate an Executive Report. The report provides HTML views of application inventories that a manager can use to assess the risks and costs of supporting the application.

## Syntax

```
ExecutiveReport Workspace Folder [Project] [Detailed]
```

### Required Parameters

Workspace

Folder

### Description

Workspace file (.rwp).

Output folder.

### Optional Parameters

Project

Detailed

### Description

Project to generate the report for.

Log file.

## Example

```
ExecutiveReport C:\Workspaces\Training.rwp C:\Executive Reports  
C:\log.txt
```

# ExportDescriptions.bj

Export object descriptions to an ERD file.

## Action

Export object descriptions to an ERD file.

## Syntax

```
ExportDescriptions Workspace ERD [Entity] [Project] [Detailed]
```

### Required Parameters

Workspace

ERD

### Description

Workspace file (.rwp).

ERD file.

### Optional Parameters

Entity

Project

Detailed

### Description

\* or entity type of the objects to export descriptions for. Default is \*.  
Use the flag attribute of an entity type to specify all entity types with that flag, for example:

```
*LEGACY
```

which specifies all entity types with the LEGACY flag. For more on flags, see *Software Development Toolkit*, available from support services.

Project to export object descriptions from.

Log file.

## Example

```
ExportDescriptions C:\Workspaces\Training.rwp C:\ObjectsERD.xml  
COBOL C:\log.txt
```

# ExportRules.bj

Export business rules to an ERD file.

## Action

Export business rules to an ERD file.

## Syntax

```
ExportRules Workspace FileName [Detailed]
```

### Required Parameters

Workspace

FileName

### Description

Workspace file (.rwp).

ERD file.

### Optional Parameters

Detailed

### Description

Log file.

## Example

```
ExportRules C:\Workspaces\Training.rwp C:\RulesERD.xml  
C:\log.txt
```

# ExportScreens.bj

Export renderings for screens in the workspace.

## Action

Export renderings for screens in the workspace.

## Syntax

```
ExportScreens Workspace [Pattern] [Output] [Project] [Detailed]
```

### Required Parameters

### Description

Workspace

Workspace file (.rwp).

### Optional Parameters

### Description

Pattern

Pattern for naming screen renderings, consisting of the output folder, file name pattern, and extension. For example, D:\*.\*.rtf. The file name pattern may contain any valid symbols. An asterisk (\*) is replaced with the name of the screen. The format of the renderings depends on the extension. Supported extensions are .rtf and .doc.

Output

Output folder if not specified in Pattern.

Project

Project to export screen renderings from.

Detailed

Log file.

## Example

```
ExportScreens C:\Workspaces\Training.rwp D:*.*.rtf C:\log.txt
```

# GenCopybooks.bj

Generate copybooks from Database Description, Device Description, or DMSII DASDL files.

## Action

Generate copybooks from Database Description, Device Description, or DMSII DASDL files. RPG programs and Cobol programs that execute in the AS/400 environment often use copy statements that reference Database Description or Device Description files rather than copybooks. MCP Cobol programs use copy statements that reference DMSII DASDL files. If your application uses copy statements to reference these types of files, you need to verify the files and generate copybooks for the application before you verify program files.

## Syntax

```
GenCopybooks Workspace [Entity] [Condition] [Convert]  
[Overwrite] [Options] [Project] [Notify] [Detailed]
```

### Required Parameters

Workspace

### Description

Workspace file (.rwp).

Optional Parameters	Description
Entity	* or entity type of source files to generate copybooks from. Default is *.
Condition	Scope of source. Use the Repository Exchange Protocol (RXP) to code the condition. For more information, see <i>Analyzing Projects</i> in the workbench documentation set. Default is project.
Convert	Specify this argument with an empty value to generate target copybooks and convert them to physical copybooks in the same step.
Overwrite	Specify this argument with an empty value to overwrite existing physical copybooks with the same name.
Options	Options script file. Default values for options usually are acceptable. Contact support services for special needs.
Project	Project to generate copybooks for.
Notify	Notification file.
Detailed	Log file.

### Example

```
GenCopybooks C:\Workspaces\Training.rwp DBFILE C:\log.txt Auto
```



# GenScreens.bj

Generate screens from Device Description files.

## Action

Generate screens from Device Description files.

## Syntax

```
GenScreens Workspace [Entity] [Condition] [Project] [Notify]
[Detailed]
```

### Required Parameters

### Description

Workspace

Workspace file (.rwp).

### Optional Parameters

### Description

Entity

\* or entity type of source files to generate screens from. Default is \*.

Condition

Scope of source. Use the Repository Exchange Protocol (RXP) to code the condition. For more information, see *Analyzing Projects* in the workbench documentation set. Default is project.

Project

Project to generate screens for.

Notify

Notification file.

Detailed

Log file.

## Example

```
GenScreens C:\Workspaces\Training.rwp DEVICEFILE C:\log.txt Auto
```

# ImpactReport.bj

Generate an Impact Subtree Report.

## Action

Generate an Impact Subtree Report. The report shows the impact trace subtree for the specified data item occurrence in XML format or in a database.

## Syntax

```
ImpactReport Workspace Entity Name HCID FileName [Direction]  
[Project] [Detailed]
```

### Required Parameters

### Description

Workspace

Workspace file (.rwp).

Entity

Entity type of the data item.

Name

Name of the data item.

HCID

HCID of the data item occurrence.  
Alternatively, use the following arguments:

- *Var*, to specify the data item name.
- *Source*, to specify the relative path of the source file containing the data item occurrence.
- *Row*, to specify the row number of the data item occurrence in the source file.
- *Col*, to specify the column number of the data item occurrence in the source file.

FileName

Output file. The format of the report depends on the extension. Supported extensions are .xml and .mdb.

### Optional Parameters

Direction

### Description

Direction of the trace:

- F or 1, to specify a forward trace.
- B or 0, to specify a backward trace.

Forward is the default.

Project

Project to generate the report for.

Detailed

Log file.

### Example

```
ImpactReport C:\Workspaces\Training.rwp variable CATALOG-MASTER
S91 C:\Impact.xml B C:\log.txt
```

# ImpactReportFromList.bj

Generate an Impact Subtree Report from a Clipper list.

## Action

Generate an Impact Subtree Report from a Clipper list. The report shows the impact trace subtrees for occurrences of data items in the list in XML format or in a database.

## Syntax

```
ImpactReportFromList Workspace Model ListName Category Output
[Direction] [Split] [Project] [Detailed]
```

### Required Parameters

Workspace

### Description

Workspace file (.rwp).

### Required Parameters

	Description
Model	HyperView model for the source files containing the data item occurrences in the list.
ListName	Name of the list.
Category	Category of the list.
Output	Output file. The format of the report depends on the extension. Supported extensions are .xml and .mdb. When Split is set to Y, the path of the folder for .mdb output files.

### Optional Parameters

	Description
Direction	Direction of the trace: <ul style="list-style-type: none"><li>• F or 1, to specify a forward trace.</li><li>• B or 0, to specify a backward trace.</li></ul> Forward is the default.
Split	Whether to use the <i>split program</i> method, Y or N. The split program method generates a separate .mdb output file for each program that contains a data item occurrence in the list. N is the default.
Project	Project to generate the report for.
Detailed	Log file.

### Example

```
ImpactReportFromList C:\Workspaces\Training.rwp COBOL Impacts  
Impact Report C:\Impacts.xml C:\log.txt
```

# ImpExBAV.bj

Import or export batch job dependencies or user names.

## Action

Import or export batch job dependencies or user names from Batch Application Viewer (BAV).

## Syntax

```
ImpExBAV Workspace Op FileName [Project] [Notify] [Detailed]
```

### Required Parameters

Workspace

### Description

Workspace file (.rwp).

Op

Operation. Specify:

- *ImportDeps*, to import dependencies.
- *ExportDeps*, to export dependencies.
- *ImportUsers*, to import user names.
- *ExportUsers*, to export user names.

FileName

Output file.

### Optional Parameters

Project

### Description

Project to import/export from/to.

Notify

Notification file.

Detailed

Log file.

## Example

```
ImpExBAV C:\Workspaces\Training.rwp ImportDeps
C:\Dependencies.xml C:\log.txt
```

# ImportRules.bj

Import business rules from an ERD file.

## Action

Import business rules from an ERD file.

## Syntax

```
ImportRules Workspace FileNames [Mode] [Detailed]
```

Required Parameters	Description
Workspace	Workspace file (.rwp).
FileNames	File name of ERD file, or a pattern to match ERD file names.

### Optional Parameters

Mode

### Description

How to handle rules that have the same internal names as existing rules. Specify:

- Creating, to create rules with unique internal names, whether or not they have the same internal names.
- Replacing, to replace existing rules whether or not they have been updated.
- Updating, to replace existing rules only when they have been updated.

The update test is against the Last Validation Time attribute. Replacing is the default.

Detailed

Log file.

### Example

```
ImportRules C:\Workspaces\Training.rwp C:\RulesERD.xml Updating
C:\log.txt
```

## IMS Analysis.bj

Perform IMS Analysis.

### Action

Perform IMS Analysis. IMS Analysis determines the types of database operation (insert, read, update, or delete) IMS programs perform, and lists in the browser each of the database segments or screens the operations are performed on.

### Syntax

```
IMS Analysis Workspace [WorkspaceWide] [Project] [Notify]
[Detailed] [Drop] [LaunchHHC] [StopHHC] [Wait]
```

**Required Parameters**

Workspace

**Description**

Workspace file (.rwp).

**Optional Parameters**

WorkspaceWide

**Description**

Whether to perform IMS Analysis across the workspace, Yes or No. Default is Yes.

Project

If WorkspaceWide is set to No, project to perform IMS Analysis for.

Notify

Notification file.

Detailed

Log file.



## Oracle Only Parameters

Drop

## Description

Whether to drop repository indexes.  
Specify:

- Auto, to let the script determine whether to drop repository indexes based on the number of files to be analyzed.
- Yes, to drop repository indexes.
- No, to not drop repository indexes.

Dropping repository indexes generally improves analysis performance when a large number of files need to be analyzed. Dropped indexes are restored when the analysis is complete.

LaunchHHC

Whether to launch the HyperCode Converter, Yes or No. Launching the HyperCode Converter generally improves performance.

StopHHC

Whether to stop the HyperCode Converter(s) when the analysis is complete, Yes or No.

Wait

Whether to wait until the HyperCode Converter(s) queue is empty. Specify:

- Yes, to wait indefinitely.
- No, to not wait.
- The number of seconds to wait for the count on the queue to change. If the count does not change within the specified time, BRP resumes. Sixty minutes is recommended. 0 means no timeout.

## Example

```
IMS Analysis C:\Workspaces\Training.rwp C:\log.txt Auto Yes Yes
3600
```

# Invalidate.bj

Invalidate source files.

## Action

Invalidate source files. Invalidating some or all of the source files in a workspace before reverifying can save time when you reverify very large workspaces.

## Syntax

```
Invalidate Workspace [Entity] [Cond] [ObjList] [Detailed] [Drop]
```

### Required Parameters

Workspace

### Description

Workspace file (.rwp).

### Optional Parameters

Entity

### Description

\* or entity type of source files to invalidate. Default is \*.

Cond

Source files to invalidate. Use the Repository Exchange Protocol (RXP) to code the condition. For more information, see *Analyzing Projects* in the workbench documentation set.

ObjList

When Cond is not set, a control file with a list of source files to invalidate. Each line of the control file contains the following information:

```
"EntityType" "EntityName"
```

where:

- *EntityType* is the entity type of the source file to invalidate, COBOL, for example.
- *EntityName* is the name of the source file to invalidate, DayOfWeek.cbl, for example.

### Optional Parameters

Detailed

### Description

Log file.

### Oracle Only Parameters

Drop

### Description

Whether to drop repository indexes.  
Specify:

- Auto, to let the script determine whether to drop repository indexes based on the number of files to be processed.
- Yes, to drop repository indexes.
- No, to not drop repository indexes.

Dropping repository indexes generally improves performance when a large number of files need to be processed. Dropped indexes are restored when processing is complete.

### Example

```
Invalidate C:\Workspaces\Training.rwp COBOL ControlFile.txt
C:\log.txt Auto
```

# InventoryReport.bj

Generate an Inventory Report.

## Action

Generate an Inventory Report. The report shows high-level statistics for source file types in the current workspace: number of files of each type, whether verified, number of lines of code (verified files only), and the like.

## Syntax

```
InventoryReport Workspace File [Project] [Detailed]
```

### Required Parameters

### Description

Workspace

Workspace file (.rwp).

File

Output file. The format of the report depends on the extension. Supported extensions are .html, .htm, .xls, .rtf, .doc, .txt, and .csv.

### Optional Parameters

### Description

Detailed

Log file.

### Example

```
InventoryReport C:\Workspaces\Training.rwp C:\Inventory.htm  
C:\log.txt
```

## Populate.bj

Populate a workspace from an ERD file.

### Action

Populate a workspace from an ERD file.

### Syntax

```
Populate Workspace ERD [Detailed]
```

### Required Parameters

Workspace

ERD

### Description

Workspace file (.rwp).

ERD file, or a file that contains a list of ERD files. If the latter, the list file name must be preceded by an @ symbol, for example, @ERDs.txt. Each line of the list file specifies the full path name of an ERD file.

### Optional Parameters

Detailed

### Description

Log file.

### Example

```
Populate C:\Workspaces\Training.rwp C:\TrainingERD.xml
C:\log.txt
```

# ReferenceReport.bj

Generate Reference Reports.

## Action

Generate Reference Reports. The reports identify missing or unneeded files or objects in the workspace:

- An *Unresolved Report* identifies missing application elements.
- An *Unreferred Report* identifies unreferenced application elements.
- A *Cross-reference Report* identifies all application references.
- An *External Reference Report* identifies references in object-oriented applications to external files that are not registered in the workspace, such as .java, Java Archive (JAR), or C++ include files (assuming you have identified the locations of these files in the Workspace Verification options

window for the source files). These references are not reported as unresolved in the Unresolved Report.

## Syntax

```
ReferenceReport Workspace Type File [Entities] [Restrict]  
[Project] [Detailed]
```

### Required Parameters

	Description
Workspace	Workspace file (.rwp).
Type	The type of report, Unresolved, Unreferred, CrossRef, or ExternalRef.
File	Output file. The format of the report depends on the extension. Supported extensions are .html, .htm, .xls, .rtf, .doc, .txt, and .csv.

### Optional Parameters

	Description
Entities	* or a comma-separated list of entity types to report on. Default is *.
Restrict	Whether to restrict references to the specified project, Yes or No. Default is Yes.
Project	Project to generate the report for. Default is the current project.
Detailed	Log file.

### Example

```
ReferenceReport C:\Workspaces\Training.rwp CrossRef  
C:\CrossRef.htm No C:\log.txt
```

# Refresh.bj

Register and verify new source files, refresh and verify updated source files.

## Action

Register and verify new source files, refresh and verify updated source files.

*NOTE: The Refresh2.bj variant also autoresolves decisions.*

## Syntax

```
Refresh Workspace StageDir [Project] [Notify] [Detailed] [Drop]
[LaunchHHC] [ExtraHHC] [StopHHC] [Wait]
```

### Required Parameters

Workspace

StageDir

### Description

Workspace file (.rwp).

Staging directory for incoming source files.

### Optional Parameters

Project

Notify

Detailed

### Description

Project to refresh.

Notification file.

Log file.

### Oracle Only Parameters

### Description

Drop

Whether to drop repository indexes.

Specify:

- Auto, to let the script determine whether to drop repository indexes based on the number of files to be processed.
- Yes, to drop repository indexes.
- No, to not drop repository indexes.

Dropping repository indexes generally improves performance when a large number of files need to be processed. Dropped indexes are restored when processing is complete.

LaunchHHC

Whether to launch the HyperCode Converter, Yes or No. Launching the HyperCode Converter generally improves performance.

ExtraHHC

If LaunchHHC is specified, the number of additional HyperCode Converters to launch.

StopHHC

Whether to stop the HyperCode Converter(s) when processing is complete, Yes or No.

Wait

Whether to wait until the HyperCode Converter(s) queue is empty. Specify:

- Yes, to wait indefinitely.
- No, to not wait.
- The number of seconds to wait for the count on the queue to change. If the count does not change within the specified time, BRP resumes. Sixty minutes is recommended. 0 means no timeout.

### Example

```
Refresh C:\Workspaces\Training.rwp E:\StagingArea C:\log.txt  
Auto Yes Yes 3600
```



# Register.bj

Register new source files, refresh updated source files.

## Action

Register new source files, refresh updated source files. Use:

- AddNew.bj to register new source files only.
- UpdateOnly.bj to refresh updated source files only.
- Refresh.bj to register and verify new and updated source files.
- Verify.bj to verify registered source files.

## Syntax

`Register Workspace StageDir [Project] [Entity] [Detailed] [Drop]`

### Required Parameters

Workspace

StageDir

### Description

Workspace file (.rwp).

Staging directory for incoming source files.

### Optional Parameters

Entity

Project

Detailed

### Description

\* or entity type of source files to register or refresh. Default is \*.

Project to register or refresh source files in.

Log file.

### Oracle Only Parameters

### Description

Drop

Whether to drop repository indexes.

Specify:

- Auto, to let the script determine whether to drop repository indexes based on the number of files to be processed.
- Yes, to drop repository indexes.
- No, to not drop repository indexes.

Dropping repository indexes generally improves performance when a large number of files need to be processed. Dropped indexes are restored when processing is complete.

### Example

```
Register C:\Workspaces\Training.rwp E:\StagingArea C:\log.txt  
Auto Yes Yes 3600
```

## Related.bj

Create a project control file (PCF) based on the relationships between source files.

### Action

Create a *project control file (PCF)* based on the relationships between source files. The source file on the left side of the relationship is called the *startup object*. The source file on the right side of the relationship is called the *target object*.

A project control file identifies the projects to which source files belong. Use `ApplyPCF.bj` or `SetProject .bj` to assign source files to projects based on a project control file.

## Syntax

```
Related Workspace Out [List] [Project] [Startup] [Target]
[Include] [Detailed]
```

Required Parameters	Description
Workspace	Workspace file (.rwp).
Out	Output file (.pcf).

Optional Parameters	Description
List	<p>A control file with a list of startup objects. Each line of the control file contains the following information:</p> <pre data-bbox="992 426 1390 455">"EntityType" "EntityName"</pre> <p>where:</p> <ul data-bbox="992 514 1463 646" style="list-style-type: none"><li>• <i>EntityType</i> is the entity type of the startup object, COBOL, for example.</li><li>• <i>EntityName</i> is the name of the startup object, DayOfWeek.cbl, for example.</li></ul>
Project	<p>When List is not specified, the project containing the startup objects.</p>
Startup	<p>When List is not specified, the entity type of the startup objects. Enclose multiple entity types in parentheses in a vertical-line-separated list, for example:</p> <pre data-bbox="992 921 1308 951">( COBOL   COPYBOOK   BMS )</pre> <p>Use the flag attribute of an entity type to specify all entity types with that flag, for example:</p> <pre data-bbox="992 1083 1105 1113">*LEGACY</pre> <p>which specifies all entity types with the LEGACY flag, the default. For more on entity flags, see <i>Software Development Toolkit</i>, available from support services.<b>NOTE:</b> <i>Additional operators are available for special needs. Contact support services for details.</i></p>
Target	<p>The entity type of the target objects. The notation is as for Startup. Default is (BMS PSB DBD CSD).</p>
Include	<p>Whether to include source files related to other source files in relationships flagged R_USE, such as Cobol Includes Copybook File. Default is Yes. Specify NONE for No. Restrict the result to source files of given types by specifying the types, for example:</p> <pre data-bbox="992 1717 1292 1747">Include= ( COBOL   JCL )</pre> <p>The notation is as for Startup. For more on relationship flags, see <i>Software Development Toolkit</i>, available from support services.</p>

Optional Parameters	Description
Detailed	Log file.

### Example

Related C:\Workspaces\Training.rwp E:\Training.pcf  
ControlFile.txt BMS C:\log.txt

# ResolveDecisions.bj

Resolve decisions automatically.

## Action

Resolve decisions automatically.

## Syntax

```
ResolveDecisions Workspace [Project] [Notify] [Detailed] [Drop]  
[LaunchHHC] [ExtraHHC] [StopHHC] [Wait]
```

Required Parameters	Description
Workspace	Workspace file (.rwp).

Optional Parameters	Description
Project	Project to resolve decisions for.
Notify	Notification file.
Detailed	Log file.

### Oracle Only Parameters

### Description

Drop

Whether to drop repository indexes.

Specify:

- Auto, to let the script determine whether to drop repository indexes based on the number of files to be processed.
- Yes, to drop repository indexes.
- No, to not drop repository indexes.

Dropping repository indexes generally improves performance when a large number of files need to be processed. Dropped indexes are restored when processing is complete.

LaunchHHC

Whether to launch the HyperCode Converter, Yes or No. Launching the HyperCode Converter generally improves performance.

ExtraHHC

If LaunchHHC is specified, the number of additional HyperCode Converters to launch.

StopHHC

Whether to stop the HyperCode Converter(s) when processing is complete, Yes or No.

Wait

Whether to wait until the HyperCode Converter(s) queue is empty. Specify:

- Yes, to wait indefinitely.
- No, to not wait.
- The number of seconds to wait for the count on the queue to change. If the count does not change within the specified time, BRP resumes. Sixty minutes is recommended. 0 means no timeout.

### Example

```
ResolveDecisions C:\Workspaces\Training.rwp C:\log.txt Auto Yes  
Yes 3600
```

# RestoreDecisions.bj

Restore resolved decisions.

## Action

Restore resolved decisions. Reverifying a file invalidates resolved decisions. Use RestoreDecisions.bj with a *decisions control file (DCF)* to restore resolved decisions. Use SaveDecisions.bj to create a decisions control file before reverifying.

## Syntax

```
RestoreDecisions Workspace DecisionsCF [Detailed]
```

### Required Parameters

Workspace

DecisionsCF

### Description

Workspace file (.rwp).

Decisions control file (DCF).

### Optional Parameters

Detailed

### Description

Log file.

## Example

```
RestoreDecisions C:\Workspaces\Training.rwp C:\ControlFile.txt  
C:\log.txt
```

# RXP.bj

Restore manually resolved decisions.

## Action

Execute a Repository Exchange Protocol (RXP) query. RXP is an XML-based API that you can use to interact with application-level information in the workspace repository. For more information, see *Analyzing Projects* in the workbench documentation set.

## Syntax

```
RXP Workspace RXP [Query] [Output] [Project] [Detailed]
```

### Required Parameters

Workspace

RXP

### Description

Workspace file (.rwp).

File that contains RXP queries.

### Optional Parameters

Query

Output

Project

Detailed

### Description

\* or name of query to execute. Default is \*.

Output file. The format of the file depends on the extension. Supported extensions are .html, .htm, .xml, .xls, .rtf, .doc, .txt, and .csv.

Project to execute queries against.

Log file.

## Example

```
RXP C:\Workspaces\Training.rwp C:\Queries.xml C:\log.txt
```



# SaveDecisions.bj

Create a decisions control file (DCF).

## Action

Create a *decisions control file (DCF)* for a workspace. A decisions control file identifies the decisions in the workspace and the objects they have been resolved to. After reverification (which invalidates decisions), use RestoreDecisions.bj to restore the resolved decisions to the workspace.

## Syntax

```
SaveDecisions Workspace DecisionsCF [Decisions] [Rels]
[Detailed]
```

Required Parameters	Description
Workspace	Workspace file (.rwp).
DecisionsCF	Output file (.txt).

### Optional Parameters

### Description

Decisions

Type of decisions to include. Specify:

- All, to include all decision types.
- Uncompleted, to include uncompleted decisions.
- Unresolved, to include unresolved decisions.

Default is Unresolved.

Rel

Whether to include relationships in the DCF, Yes or No. Default is No.

Detailed

Log file.

### Example

```
SaveDecisions C:\Workspaces\Training.rwp C:\ControlFile.txt All  
Yes C:\log.txt
```

## SetProject.bj

Assign source files to projects based on a project control file (PCF).

### Action

Assign source files to projects based on a *project control file (PCF)*. A project control file identifies the projects to which source files belong.

SetProject.bj differs from ApplyPCF.bj in that it allows you to assign source files to projects additively, without deleting their links to existing projects. Use CreatePCF.bj or Related.bj to create a project control file.

### Syntax

```
SetProject Workspace ProjectCF [Incremental] [Detailed]
```

#### Required Parameters

Workspace

ProjectCF

#### Description

Workspace file (.rwp).

Project control file (.pcf).

#### Optional Parameters

Incremental

Detailed

#### Description

Assign source files to projects additively, without deleting their links to existing projects. Use the argument with no value to specify additive assignment. Omit the argument to specify overwrite assignment.

Log file.

#### Example

```
SetProject C:\Workspaces\Training.rwp E:\Training.pcf
Incremental C:\log.txt
```

## Unregister.bj

Unregister source files.

### Action

Unregister source files.

### Syntax

```
Unregister Workspace [Entity] [Cond] [ObjList] [Detailed] [Drop]
```

**Required Parameters**

Workspace

**Description**

Workspace file (.rwp).

**Optional Parameters**

Entity

**Description**

\* or entity type of source files to unregister. Default is \*.

Cond

Source files to unregister. Use the Repository Exchange Protocol (RXP) to code the condition. For more information, see *Analyzing Projects* in the workbench documentation set.

ObjList

When Cond is not set, a control file with a list of source files to unregister. Each line of the control file contains the following information:

*"EntityType" "EntityName"*

where:

- *EntityType* is the entity type of the source file to unregister, COBOL, for example.
- *EntityName* is the name of the source file to unregister, DayOfWeek.cbl, for example.

Detailed

Log file.

### Oracle Only Parameters

Drop

### Description

Whether to drop repository indexes.

Specify:

- Auto, to let the script determine whether to drop repository indexes based on the number of files to be processed.
- Yes, to drop repository indexes.
- No, to not drop repository indexes.

Dropping repository indexes generally improves performance when a large number of files need to be processed. Dropped indexes are restored when processing is complete.

### Example

```
Unregister C:\Workspaces\Training.rwp COBOL ControlFile.txt
C:\log.txt Auto
```

## UpdateOnly.bj

Refresh updated source files only.

### Action

Refresh updated source files only, optionally based on a project control file (PCF). A project control file identifies the projects to which source files belong.

Use:

- CreatePCF.bj or Related.bj to create a project control file.
- Register.bj to register new source files and refresh updated source files.
- Refresh.bj to register and verify new and updated source files.
- Verify.bj to verify registered source files.

## Syntax

```
UpdateOnly Workspace StageDir [ProjectCF] [Notify] [Detailed]  
[Drop]
```

### Required Parameters

Workspace

StageDir

### Description

Workspace file (.rwp).

Staging directory for incoming source files.

### Optional Parameters

ProjectCF

Notify

Detailed

### Description

Project control file (PCF).

Notification file.

Log file.

### Oracle Only Parameters

Drop

### Description

Whether to drop repository indexes.  
Specify:

- Auto, to let the script determine whether to drop repository indexes based on the number of files to be processed.
- Yes, to drop repository indexes.
- No, to not drop repository indexes.

Dropping repository indexes generally improves performance when a large number of files need to be processed. Dropped indexes are restored when processing is complete.

## Example

```
UpdateOnly C:\Workspaces\Training.rwp E:\StagingArea  
C:\Training.txt C:\log.txt Auto
```

# Upgrade.bj

Upgrade a workspace.

## Action

Upgrade a workspace. Upgrading a workspace synchronizes the workspace with a new MW configuration.

## Syntax

`Upgrade Workspace [Detailed]`

### Required Parameters

Workspace

### Description

Workspace file (.rwp).

### Optional Parameters

Detailed

### Description

Log file.

## Example

`Upgrade C:\Workspaces\Training.rwp C:\log.txt`

# Verify.bj

Verify source files.

## Action

Verify registered source files. Use Refresh to register and verify new and updated source files.

## Syntax

```
Verify Workspace [Entity] [Status] [Cond] [Project] [Notify]  
[Detailed] [Drop] [LaunchHHC] [ExtraHHC] [StopHHC] [Wait]
```

### Required Parameters

Workspace

### Description

Workspace file (.rwp).

### Optional Parameters

Entity

\* or entity type of source files to verify.  
Default is \*.

Status

Verification status of source files to verify.  
Specify:

- \*, to verify all source files.
- !, to verify unverified source files.
- R, to verify source files verified under the relaxed parsing option.
- E, to verify source files that verified with an error.
- S, to verify source files that verified successfully.

Cond

Source files to verify. Use the Repository Exchange Protocol (RXP) to code the condition. For more information, see *Analyzing Projects* in the workbench documentation set.



Optional Parameters	Description
Project	Project to verify source files in.
Notify	Notification file.
Detailed	Log file.

### Oracle Only Parameters

### Description

Drop

Whether to drop repository indexes.

Specify:

- Auto, to let the script determine whether to drop repository indexes based on the number of files to be processed.
- Yes, to drop repository indexes.
- No, to not drop repository indexes.

Dropping repository indexes generally improves performance when a large number of files need to be processed. Dropped indexes are restored when processing is complete.

LaunchHHC

Whether to launch the HyperCode Converter, Yes or No. Launching the HyperCode Converter generally improves performance.

ExtraHHC

If LaunchHHC is specified, the number of additional HyperCode Converters to launch.

StopHHC

Whether to stop the HyperCode Converter(s) when processing is complete, Yes or No.

Wait

Whether to wait until the HyperCode Converter(s) queue is empty. Specify:

- Yes, to wait indefinitely.
- No, to not wait.
- The number of seconds to wait for the count on the queue to change. If the count does not change within the specified time, BRP resumes. Sixty minutes is recommended. 0 means no timeout.

### Example

```
Verify C:\Workspaces\Training.rwp COBOL R C:\log.txt Auto Yes  
Yes 3600
```

# Executing Batch Scripts

Overview of the Batch Refresh and Verification (Brave) utility.

Use the Batch Refresh and Verification (Brave) utility to execute batch scripts. Brave.exe is located in \<Workbench Home>\Bin.

The examples in this section illustrate how to run the scripts with Brave.exe. The examples can be adapted for use programmatically or in a batch file.

## Example: Generating Reports

Describes how to generate an Unresolved Report in batch mode.

Follow the steps below to generate an Unresolved Report in Excel format. Refer to ReferenceReport.bj for argument details.

- 1) From a command prompt, enter the following command, substituting file names and paths as appropriate:

```
C:\Program Files\Modernization Workbench\Bin>Brave.exe
"C:\Program Files\Moderni
zation Workbench\Scripts\BRP\ReferenceReport.bj"
"C:\UnresolvedLog.txt" "Workspa
ce=C:\Workspaces\sdkworkspace.rwp" "Type=Unresolved"
"File=C:\Workspaces\sdkwork
space\Output\UnresolvedReport.xls"
```

The command consists of:

- The path to Brave.exe.
- The path to the ReferenceReport.bj file.
- The path to the output log file generated on execution of the command.
- The path to the workspace.
- The type of reference report to generate.
- The path to the output report. The format of the report depends on the extension.

- 2) Check the output log file for errors or warnings. Here is the log file for the command:

```
Batch Registration and Verification. Version 2.1.02.2860 (build
2.1.02.2860)
Date: 8/8/2008 Computer: D620-JEREMYW
```

```
Cmd:  "C:\Program Files\Modernization
Workbench\Scripts\BRP\ReferenceReport.bj"
"C:\UnresolvedLog.txt"
"Workspace=C:\Workspaces\sdkworkspace.rwp" "Type=Unresolved"
"File=C:\Workspaces\sdkworkspace\Output\UnresolvedReport.xls"
Job: C:\Program Files\Modernization
Workbench\Scripts\BRP\ReferenceReport.bj
13:43:13 >Open C:\Workspaces\sdkworkspace.rwp
13:43:15 >Report Unresolved
C:\Workspaces\sdkworkspace\Output\UnresolvedReport.xls
13:43:23 C:\Workspaces\sdkworkspace\Output\UnresolvedReport.xls
has been prepared
13:43:23 >Close
13:43:24 ---Finished---
```

## Example: Executing Repository Queries

Describes how to execute an RXP query in batch mode

Follow the steps below to execute a Repository Exchange Protocol (RXP) query. RXP is an XML-based API that you can use to interact with application-level information in the workspace repository. Refer to RXP.bj for argument details.

- 1) From a command prompt, enter the following command, substituting file names and paths as appropriate:

```
C:\Program Files\Modernization Workbench\Bin>Brave.exe
"C:\Program Files\Moderni
zation Workbench\Scripts\BRP\RXP.bj" "C:\QueryLog.txt"
"Workspace=C:\Workspaces\
sdkworkspace.rwp" "RXP=C:\Program Files\Modernization
Workbench\Scripts\BRP\RXP\
Repository.rxp" "Query=Used Sources"
"Output=C:\Workspaces\sdkworkspace\Output\U
sedSources.xml"
```

The command consists of:

- The path to Brave.exe.
- The path to the RXP.bj file.
- The path to the output log file generated on execution of the command.
- The path to the workspace.
- The path to the .rxp file containing RXP queries.
- The query to execute in the .rxp file, "Used Sources".
- The path to the output file. The format of the file depends on the extension.

- 2) Check the output log file for errors or warnings. Here is the log file for the command:

```
Batch Registration and Verification. Version 2.1.02.2860 (build
2.1.02.2860)
Date: 8/8/2008      Computer: D620-JEREMYW
Cmd:  "C:\Program Files\Modernization
Workbench\Scripts\BRP\RXP.bj" "C:\QueryLog.txt"
"Workspace=C:\Workspaces\sdkworkspace.rwp" "RXP=C:\Program
Files\Modernization Workbench\Scripts\BRP\RXP\Repository.rxp"
"Query=Used Sources"
"Output=C:\Workspaces\sdkworkspace\Output\UsedSources.xml"
Job: C:\Program Files\Modernization
Workbench\Scripts\BRP\RXP.bj
14:03:32 >Open C:\Workspaces\sdkworkspace.rwp
14:03:33 Cuter .ExecuterXP (Prm.RXP, Prm.Query, Prm.Output,
Prm.Project)
14:03:34 File C:\Workspaces\sdkworkspace\Output\UsedSources.xml
has been prepared
14:03:34 >Close
14:03:34 ---Finished---
```

## Example: Creating Diagrams

Describes how to generate Call Map diagrams in batch mode.

Follow the steps below to generate Call Map diagrams in EMF format for every program in a workspace. Refer to DiagramTS.bj for argument details.

- 1) From a command prompt, enter the following command, substituting file names and paths as appropriate:

```
C:\Program Files\Modernization Workbench\Bin>Brave.exe
"C:\Program Files\Modernization Workbench\Scripts\BRP\DiagramTS.bj" "C:\DiagramLog.txt"
"Workspace=C:\
Workspaces\sdkworkspace.rwp" "Scope=Call Map"
"Pattern=C:\Workspaces\sdkworkspac
e\Output\*.emf"
```

The command consists of:

- The path to Brave.exe.
- The path to the DiagramTS.bj file.
- The path to the output log file generated on execution of the command.
- The path to the workspace.
- The diagram scope, "Call Map".

- The pattern for naming the generated diagrams, consisting of the output folder, file name pattern, and extension. The format of the diagrams depends on the extension.
- 2) Check the output log file for errors or warnings. Here is the log file for the command:

```
Batch Registration and Verification. Version 2.1.02.2860 (build
2.1.02.2860)
Date: 8/8/2008 Computer: D620-JEREMYW
Cmd: "C:\Program Files\Modernization
Workbench\Scripts\BRP\DiagramTS.bj" "C:\DiagramLog.txt"
"Workspace=C:\Workspaces\sdkworkspace.rwp" "Scope=Call Map"
"Pattern=C:\Workspaces\sdkworkspace\Output\*.emf"
Job: C:\Program Files\Modernization
Workbench\Scripts\BRP\DiagramTS.bj
13:22:41 >Open C:\Workspaces\sdkworkspace.rwp
13:22:41 >Diagram Quick * "Call Map"
"C:\Workspaces\sdkworkspace\Output\*.emf"
Destination directory is
C:\Workspaces\sdkworkspace\Output
Diagrams have been generated successfully
13:23:06 >Close
13:23:06 ---Finished---
```

## Example: Performing an Advanced Search

Describes how to perform an advanced search in batch mode.

Follow the steps below to search for all declarations of computational data items in a workspace. Refer to ClipperSearch.bj for argument details.

- 1) From a command prompt, enter the following command, substituting file names and paths as appropriate:

```
C:\Program Files\Modernization Workbench\Bin>Brave.exe
"C:\Program Files\Moderni
zation Workbench\Scripts\BRP\ClipperSearch.bj"
"C:\ClipperSearchLog.txt" "Worksp
ace=C:\Workspaces\sdkworkspace.rwp" "Criteria=General:Data
Queries\Computational
Data" "Model=COBOL" "ListName=Miscellaneous" "Category=General"
```

The command consists of:

- The path to Brave.exe.
- The path to the ClipperSearch.bj file.
- The path to the output log file generated on execution of the command.

- The path to the workspace.
  - The path to the search criterion in the HyperView Advanced Search tool, including the tab name and folder names.
  - The HyperView model for the source files to be searched, "COBOL".
  - The Clipper list where the search results will be displayed.
  - The Clipper category that contains the list.
- 2) Check the output log file for errors or warnings. Here is the log file for the command:

```
Batch Registration and Verification. Version 2.1.02.2860 (build
2.1.02.2860)
Date: 8/8/2008      Computer: D620-JEREMYW
Cmd:  "C:\Program Files\Modernization
Workbench\Scripts\BRP\ClipperSearch.bj"
"C:\ClipperSearchLog.txt"
"Workspace=C:\Workspaces\sdkworkspace.rwp"
"Criteria=General>Data Queries\Computational Data"
"Model=COBOL" "ListName=Miscellaneous" "Category=General"
Job: C:\Program Files\Modernization
Workbench\Scripts\BRP\ClipperSearch.bj
10:33:25 >Open C:\Workspaces\sdkworkspace.rwp
10:33:26 Cuter .ClipperSearch (Prm.Criterion, Prm.Model,
Prm.ListName, Prm.Category, Prm.Accumulate)
10:33:27 (success) 236 construct(s) found.
10:33:27 >Close
10:33:27 ---Finished---
```

