



## **CICS TS for z/OS: WSBInd File Display and Change Utility**

*Version V1.0.3*

---

***April 29<sup>th</sup>, 2011***

**First Edition (January 2008)** – Original SupportPac CS04 release V1.0.1

**Second Edition (November 2009)** – SupportPac CS04 release V1.0.2.

**Third Edition (April 2011)** – SupportPac CS04 release V1.0.3

© Copyright International Business Machines Corporation 2007, 2011. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

**Note:** Before using this information and the product it supports, read the information in “Notices” on page 4.

# Contents

CICS TS for z/OS: WSBInd File Display and Change Utility ..... 1

Contents ..... 3

Notices ..... 4

Trademarks ..... 5

Section 1: Overview ..... 6

Section 2: SupportPac Requirements ..... 7

Section 3: Control Statement Usage ..... 8

Section 4: Example Input and Output ..... 12

Section 5: Installation ..... 17

Section 6: Using the CS04 WSBInd File Utility ..... 18

Section 7: How the Program Supplied with CS04 Works ..... 19

Section 8: References ..... 22

## Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.*

*The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:* INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

## Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AD/Cycle®

AFS®

AIX®

C/370™

CICS®

CICSplex®

COBOL/370™

DB2®

DFS™

ETE™

eServer™

 eServer®

 eServer®

HiperSockets™

IMS™

Language Environment®

Multiprise®

MVS™

MVS/ESA™

OS/2®

OS/390®


OS/400®

Parallel Sysplex®

PR/SM™

Rational®

Redbooks™

Redbooks (logo) ™

RACF®

S/390®

VisualAge®

VTAM®

WebSphere®

z/Architecture™

z/OS®

zSeries®

z/VM®

The following terms are trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

## Section 1: Overview

The purpose of this SupportPac is to provide a way for CICS customers (CICS TS V3.2 and higher) to display and change WSBInd file contents via z/OS batch Jobs. The Java program included with this SupportPac provides a z/OS batch interface to the Java WSBInd file support classes supplied with CICS TS V3.2 and higher. The CICS-supplied Java classes do the actual reading/changing/writing of the WSBInd file values. You must, therefore, have a license for CICS TS V3.2 or higher to use this SupportPac.

To run this SupportPac, you use the JCL, PROC, script, and Java program that are provided with this SupportPac. You construct 'control statements' to tell the Java program what to do. You include the control statements with the supplied JCL (which executes the PROC, script, and Java program). In addition to the control statements, you will have one or more WSBInd files as input. The actions taken by the Java program will be listed in your Job output. Additionally, depending on your control statements, the Java program can produce updated WSBInd file(s).



Batch control statements (control cards) are available to:

- Display the characteristics of an individual WSBInd file
- Display the characteristics of all WSBInd files in a specified directory
- Display the characteristics of all WSBInd files in the specified directory and every subordinate directory
- Change some of the characteristics of individual WSBInd files

The values allowed to be changed by the Java program supplied with this SupportPac are those values allowed to be changed by the CICS TS V3.2 or later supplied Java classes that manipulate the WSBInd file.

For a Provider WSBInd file, the following values may be changed:

- URI
- Target Program Name
- Transaction Id
- User Id
- Request Channel Name
- Request Channel File Name
- Response Channel Name
- Response Channel File Name
- Sync On Return
- WSDL File Name
- WSDL 2.0 File Name
- Vendor Converter Program Name

For a Requester WSBInd file, the following values may be changed:

- Endpoint URI
- Request Channel Name
- Request Channel File Name
- Response Channel Name
- Response Channel File Name
- Vendor Converter Program Name

## Section 2: SupportPac Requirements

The Java program supplied with this SupportPac requires Java SDK 1.4.2 or higher. Additionally, this SupportPac uses Java classes supplied with CICS TS V3.2 (or later) so any restrictions that apply to those classes (e.g. required Java level) also apply to this SupportPac. Since this SupportPac uses CICS TS V3.2 (or later) supplied Java classes, you will need to have access to a CICS TS V3.2 (or later) installation, and will need appropriate CICS licensing.

This SupportPac allows you to display values or change values in a CICS WSBInd file, so some understanding of WSBInd file contents is required.

To execute the Java program in this SupportPac, you submit JCL (as detailed in later sections of this document) containing control statements that tell the program what to do. To successfully utilize the program, you will need to be able to construct control statements using some editor, and be able to submit the batch Job. The output of the supplied program is information sent to the output listing, and optionally, new or updated WSBInd files. You will therefore need a way to look at your Job output.

As part of the installation, you will need to know how to copy/FTP files to z/OS, and have the appropriate permissions to do so. Additionally, you will need to have appropriate security to change the permission on the script file you transfer to z/OS.

## Section 3: Control Statement Usage

The Java program in this SupportPac is invoked via a batch Job. You submit JCL containing control statements that tell the program what to do. The order of the control statements is important. Requests are processed in the order they occur. Two passes are performed on the control statements. The first pass looks for errors in the control statements such as the control statement sequence and some file permissions. If an error was detected on the first pass, the program immediately stops after displaying an error message. The second pass performs the requested functions. The purpose for two passes is to attempt to avoid situations where 'some' of the requests were processed and some requests were not processed. Although the program detects errors on the first pass, there are a few error situations that will occur only on the second pass. These error situations should be rare (e.g. I/O error writing the file), but if errors are detected on the second pass, the program will immediately stop after displaying an error message. In this case, you will need to look at the output to determine which if any requests were not processed.

Odd situations: Because of the two-pass operation, there are (somewhat illogical) combinations of control statements that can fool the utility. If you create a new version of a WSBInd file and then try to update the newly-created version in the same set of control statements, since no files are written on the first pass, when the utility tries to update the not-yet-written WSBInd file, the utility will produce a 'file not found' message and stop. This may seem like an illogical message, but if you understand the two-pass nature of the utility and understand that nothing is written on the first pass, the error message makes sense.

Below are some usage examples, and then a discussion of all control statements, what they do, and in what order they can occur.

JCL to invoke the program will be discussed in a different section. Additionally, installation of this SupportPac will be discussed in a different section.

### Control Statement Details:

Note that the control statements can be in upper, low, or mixed case, and are order dependent.

**DIRECTORY=** Used to specify a directory. This directory will be used when a WSBInd file is specified that is not fully qualified. This statement essentially establishes a default directory, and is valid anywhere in the control statements. This statement can be used multiple times; they are processed in the order they occur.

**DISPLAY=** Displays the contents of the specified WSBInd file. This statement is valid only before an INPUT statement. If you do not specify a file suffix, .wsbind will be added to all files that do not already end in .wsbind. See the SUFFIX command if your WSBInd files contain a suffix of something other than .wsbind.



**ENDPOINT=** Used to specify a new endpoint. This statement is valid only when the WSBInd file is for a service requestor. This statement is valid anywhere after an INPUT statement.

**FILENAMEEXPRESSION=** Used to specify a filter for WSBInd file names when using the SHOWALL, SHOWALLINDIRECTORY, RECURSEALL, SHOWALLINDIRECTORYTREE, or SHOWALLINDIRECTORYSTRUCTURE commands. This control statement allows you to use a regular expression to select which WSBInd files will be displayed. This only applies to the first part of the WSBInd file name and does not apply to the WSBInd file name suffix (see SUFFIX command). Note that the letters specified in the command are case sensitive. For more information on how to compose a regular expression, see the Java documentation pertaining to regular expressions. This command cannot be continued (i.e. ending it with a '\*' will not cause it to be a continued line). If this control statement is not used, 'all WSBInd files' is assumed for the SHOWALL, SHOWALLINDIRECTORY, RECURSEALL, SHOWALLINDIRECTORYTREE, or SHOWALLINDIRECTORYSTRUCTURE control statements.

**FILENAMEFILTER=** Used to specify a filter for WSBInd file names when using the SHOWALL, SHOWALLINDIRECTORY, RECURSEALL, SHOWALLINDIRECTORYTREE, or SHOWALLINDIRECTORYSTRUCTURE commands. This statement allows you to use a '\*' (any number of any characters) and a '?' (one occurrence of any character) to select which WSBInd files will be displayed. This statement only applies to the first part of the WSBInd file name and does not apply to the suffix (see SUFFIX command). Note that the letters specified in the command are case sensitive. This command cannot be continued (i.e. ending it with a '\*' will not cause it to be a continued line). If this control statement is not used, 'all WSBInd files' is assumed for the SHOWALL, SHOWALLINDIRECTORY, RECURSEALL, SHOWALLINDIRECTORYTREE, or SHOWALLINDIRECTORYSTRUCTURE control statements.

**HELP** - Displays the version number of the Java program, plus some help text. If HELP is requested, the HELP text will be shown on the first pass and the program will terminate, performing no actions other than listing HELP information.

**INPUT=** Used to specify the name of the input WSBInd file. NOTE: that if the specified file name does not contain a suffix (i.e. doesn't contain a '.'), a .wsbind suffix will be added to the specified name. See the SUFFIX command if you want a suffix other than .wsbind to be assumed.

**OUTPUT=** Used to specify the name of the output WSBInd file. The INPUT and the OUTPUT may be the same, in which case the INPUT file will be overlaid. NOTE: that if the specified file name does not contain a suffix (i.e. doesn't contain a '.'), a .wsbind suffix will be added. See the SUFFIX command if you want a suffix other than .wsbind to be assumed.

**PGMNAME=** Used to add or change a target program name in a WSBind file. Valid only when the WSBind file is for a service provider. This statement is valid anywhere after an INPUT statement.

**REQUESTCHANNELNAME=** Used to change a request channel name in a WSBind file. Valid only when the WSBind file is at mapping level 3.0 or higher. This statement is valid anywhere after an INPUT statement.

**REQUESTCHANNELFILENAME=** Used to change a request channel file name in a WSBind file. Valid only when the WSBind file is at mapping level 3.0 or higher. This statement is valid anywhere after an INPUT statement.

**RESPONSECHANNELNAME=** Used to change a response channel name in a WSBind file. Valid only when the WSBind file is at mapping level 3.0 or higher. This statement is valid anywhere after an INPUT statement.

**RESPONSECHANNELFILENAME=** Used to change a response channel file name in a WSBind file. Valid only when the WSBind file is at mapping level 3.0 or higher. This statement is valid anywhere after an INPUT statement.

**SHOWALLINDIRECTORYTREE=**  
**SHOWALLINDIRECTORYSTRUCTURE=**  
**RECURSEALL=**

Used to indicate a directory. This directory will be used as a starting point for a recursive directory scan (i.e. the program looks for files in this directory and any subordinate directories). Any .wsbind file in the specified directory or any recursed directory will have its details displayed. This statement must be before any INPUT statements. See the SUFFIX command if your WSBind files contain a suffix other than .wsbind. If you only want selected WSBind files to be displayed, you can use the FILENAMEFILTER or FILENAMEEXPRESSION statement prior to this statement.

**SHOWALLINDIRECTORY=**  
**SHOWALL=**

Used to specify a directory. The details for all .wsbind files in this directory will be displayed. This statement must be before any INPUT statements. See the SUFFIX command if your WSBind files contain a suffix other than .wsbind. If you only want selected WSBind files to be displayed, you can use the FILENAMEFILTER or FILENAMEEXPRESSION statement prior to this statement.

**SUFFIX=** In all other commands, when a file name is specified without a suffix, by default, a suffix of .wsbind will be added. If your files contain a suffix of something other than .wsbind, use the SUFFIX command to indicate what suffix should be added by this utility. When specifying a suffix, do not include the '.'. The suffix command can appear anywhere in the input control statements, and it can be used multiple times. Commands are addressed in the order in which they occur.

**SYNCONRETURN=** Used to indicate if 'Sync On Return' in the WSBInd file should be set to 'true' or 'false'. Any value other than 'true' or 'false' will cause the command to fail. If your dfjwsdl.jar does not contain the level of CICS-supplied Java classes that support setting a Sync On Return Value, the command will fail and the utility will stop without setting any values.

**TRANSACTION=** Used to add or change a transaction name. Valid only when the WSBInd file is for a service provider. This statement is valid anywhere after an INPUT statement.

**URI=** Used to change the path information. Valid only when the WSBInd file is for a service provider. This statement is valid anywhere after an INPUT statement.

**USERID=** Used to add or change a userid. Valid only when the WSBInd file is for a service provider. This statement is valid anywhere after an INPUT statement.

**VENDORCONVERTERPROGRAMNAME=** Used to change a vendor converter program name. Valid for either a service provider or service requester WSBInd file. This statement is valid anywhere after an INPUT statement. If the WSBInd file to be changed specifies 'Interpretive XML Conversion', an error message will be produced and the Java program will stop.

**WSDLFILENAME=** Used to indicate the WSDL file name to be placed in the WSBInd file. The value specified is treated like a string and is just placed in the WSBInd file. No checks are performed to see if the specified file name exists. If your dfjwsdl.jar does not contain the level of CICS-supplied Java classes that support setting the WSDLFileName, the command will fail and the utility will stop without setting any values.

**WSDL20FILENAME=** Used to indicate the WSDL 2.0 file name to be placed in the WSBInd file. The value specified is treated like a string and is just placed in the WSBInd file. No checks are performed to see if the specified file name exists. If your dfjwsdl.jar does not contain the level of CICS-supplied Java classes that support setting the WSDLFileName, the command will fail and the utility will stop without setting any values.

**Note:**

- Lines that start with a # are considered comment statements
- If you need to continue a line, end it with an \*, then continue the line in column 1 of the next statement (same as the DFHWS2LS utility). There are two exceptions; the FILENAMEFILTER and FILENAMEEXPRESSION statements cannot be continued.

## Section 4: Example Input and Output

This section contains 4 example usages of the WSBIND File Display and Change Utility.

- **Example 1** takes a requester WSBIND file as input and creates a WSBIND file for a test, QA, and production environment. Each of the three output WSBIND files are the same as the input, except that they have different endpoints.
- **Example 2** changes the TRANSACTION in a WSBIND file.
- **Example 3** displays the contents of a single WSBIND file.
- **Example 4** displays selected WSBIND files in a directory tree. The files are selected by specifying a file filter. In this case we want to show all WSBIND files that start with a lowercase "f" and end in a lowercase "t".

### Example 1:

The control statements below change the endpoint for a requester WSBIND file for test, QA, and production environments. In addition to the Job output shown below, you would have fundProgOutTest.wsbind, fundProgOutQA.wsbind, and fundProgOutProd.wsbind files.

#### Input:

```
#-> make copies with different endpoints
Directory=/u/team10/cicslab/wsbind
Input=fundProgOut

Endpoint=http://somehost2:12111/test/someService
Output=fundProgOutTest

Endpoint=http://somehost3:13111/qa/someService
Output=fundProgOutQa

Endpoint=http://somehost4:14111/prod/someService
Output=fundProgOutProd
```

#### Output:

```
==> WSBIND File Display and Change Utility starting (V1.0.3)
==> Collecting statements. Statements will be scanned twice. If there are any
detected problems on the first pass, no actions will be taken.
==> -> stmt 1, comment => #-> make copies with different endpoints
==> -> stmt 2, =====> Directory=/u/team10/cicslab/wsbind
==> -> stmt 3, =====> Input=fundProgOut
==> -> stmt 4, blank =>
==> -> stmt 5, =====> Endpoint=http://somehost2:12111/test/someService
==> -> stmt 6, =====> Output=fundProgOutTest
==> -> stmt 7, blank =>
==> -> stmt 8, =====> Endpoint=http://somehost3:13111/qa/someService
==> -> stmt 9, =====> Output=fundProgOutQa
==> -> stmt 10, blank =>
==> -> stmt 11, =====> Endpoint=http://somehost4:14111/prod/someService
==> -> stmt 12, =====> Output=fundProgOutProd

==> ==> Starting pass 1 (a syntax check).
```

```
==> ==> Starting pass 2 (taking actions).

==> -> statement 2 => Directory=/u/team10/cicslab/wsbind
==> Default directory changed from "" to "/u/team10/cicslab/wsbind/"

==> -> statement 3 => Input=fundProgOut
==> Input WSBind file name set to /u/team10/cicslab/wsbind/fundProgOut.wsbind
Details for /u/team10/cicslab/wsbind/fundProgOut.wsbind
- Creation Timestamp: 20071024 12:47
- XML Conversion: Interpretive XML Conversion
- Mapping Level: 2.0
- Minimum Runtime Level: 2.0
- Requester
- Endpoint URI: http://zserveros.dfw.ibm.com:9010/cicslab/fundProg
- WSDL File Name: C:\workspaces\cicsws\CICSRequestor\fundProgOut.wsdl
- Operation: FUNDPROGOperation
- WSDL Binding: FUNDPROGHTTPSoapBinding

==> -> statement 5 => Endpoint=http://somehost2:12111/test/someService
==> => The old value for Endpoint URI was
"http://zserveros.dfw.ibm.com:9010/cicslab/fundProg"
==> => The new value for Endpoint URI is "http://somehost2:12111/test/someService"

==> -> statement 6 => Output=fundProgOutTest
==> Output WSBind file name set to /u/team10/cicslab/wsbind/fundProgOutTest.wsbind
==> WSBind File was written to /u/team10/cicslab/wsbind/fundProgOutTest.wsbind

==> -> statement 8 => Endpoint=http://somehost3:13111/qa/someService
==> => The old value for Endpoint URI was
"http://somehost2:12111/test/someService"
==> => The new value for Endpoint URI is "http://somehost3:13111/qa/someService"

==> -> statement 9 => Output=fundProgOutQa
==> Output WSBind file name set to /u/team10/cicslab/wsbind/fundProgOutQa.wsbind
==> WSBind File was written to /u/team10/cicslab/wsbind/fundProgOutQa.wsbind

==> -> statement 11 => Endpoint=http://somehost4:14111/prod/someService
==> => The old value for Endpoint URI was "http://somehost3:13111/qa/someService"
==> => The new value for Endpoint URI is "http://somehost4:14111/prod/someService"

==> -> statement 12 => Output=fundProgOutProd
==> Output WSBind file name set to /u/team10/cicslab/wsbind/fundProgOutProd.wsbind
==> WSBind File was written to /u/team10/cicslab/wsbind/fundProgOutProd.wsbind

==> WSBind File Display and Change Utility ending
```

**Example 2:**

Example 2: The control statements below change the TRANSACTION in a WSBind file. At the end of the Job, you would have an updated fundProg.wsbind file. The output WSBind file could have been a new WSBind file, but in this case, we just updated the existing WSBind file. You do not have to use the directory statement, you can fully qualify your input and output.

**Input:**

```
DIRECTORY=/u/team10/cicslab/wsbind
INPUT=fundProg.wsbind
TRANSACTION=DDW0
OUTPUT=fundProg.wsbind
```

**Output:**

```
==> WSBind File Display and Change Utility starting (V1.0.3)
==> Collecting statements. Statements will be scanned twice. If there are any
detected problems on the first pass, no actions will be taken.
==> -> stmt 1, =====> DIRECTORY=/u/team10/cicslab/wsbind
==> -> stmt 2, =====> INPUT=fundProg.wsbind
==> -> stmt 3, =====> TRANSACTION=DDW0
==> -> stmt 4, =====> OUTPUT=fundProg.wsbind

==> ==> Starting pass 1 (a syntax check).
==> ==> Starting pass 2 (taking actions).

==> -> statement 1 => DIRECTORY=/u/team10/cicslab/wsbind
==> Default directory changed from "" to "/u/team10/cicslab/wsbind/"

==> -> statement 2 => INPUT=fundProg.wsbind
==> Input WSBind file name set to /u/team10/cicslab/wsbind/fundProg.wsbind
Details for /u/team10/cicslab/wsbind/fundProg.wsbind
- Creation Timestamp: 20071019 14:50
- XML Conversion: Interpretive XML Conversion
- Mapping Level: 1.0
- Minimum Runtime Level: 1.0
- Provider
- Target program name: FUNDPROG
- Program interface: COMMAREA
- Transaction ID: DDW1
- User ID: -none specified-
- URI: /cicslab/fundProg
- WSDL File Name: /u/team10/cicslab/wsdl/fundProg.wsdl
- Operation: FUNDPROGOperation
- WSDL Binding: FUNDPROGHTTPSoapBinding

==> -> statement 3 => TRANSACTION=DDW0
==> => The old value for Transaction Id was "DDW1"
==> => The new value for Transaction Id is "DDW0"

==> -> statement 4 => OUTPUT=fundProg.wsbind
==> Output WSBind file name set to /u/team10/cicslab/wsbind/fundProg.wsbind
==> WSBind File /u/team10/cicslab/wsbind/fundProg.wsbind already exists, but will
be overwritten...
==> WSBind File was written to /u/team10/cicslab/wsbind/fundProg.wsbind

==> WSBind File Display and Change Utility ending
```

**Example 3:**

The control statements below display values in a single WSBind file. The input and output are shown below.

**Input:**

```
#-> Display the contents of one WSBind file
DISPLAY=/u/team10/cicslab/wsbind/fundProgOut.wsbind
```

Note that we could have also used the following statements to do the same thing:

```
Directory=/u/team10/cicslab/wsbind
Display=fundProgOut
```

**Output:**

```
==> WSBind File Display and Change Utility starting (V1.0.3)
==> Collecting statements. Statements will be scanned twice. If there are any
detected problems on the first pass, no actions will be taken.
==> -> stmt 1, comment => #-> Display the contents of one WSBind file
==> -> stmt 2, =====> DISPLAY=/u/team10/cicslab/wsbind/fundProgOut.wsbind

==> ==> Starting pass 1 (a syntax check).

==> ==> Starting pass 2 (taking actions).

==> -> statement 2 => DISPLAY=/u/team10/cicslab/wsbind/fundProgOut.wsbind
==> DISPLAY WSBind file name set to /u/team10/cicslab/wsbind/fundProgOut.wsbind
Details for /u/team10/cicslab/wsbind/fundProgOut.wsbind
- Creation Timestamp: 20071024 12:47
- XML Conversion: Interpretive XML Conversion
  - Mapping Level: 2.0
  - Minimum Runtime Level: 2.0
- Requester
  - Endpoint URI: http://zserveros.dfw.ibm.com:9010/cicslab/fundProg
- WSDL File Name: C:\workspaces\cicsws\CICSRequestor\fundProgOut.wsdl
- Operation: FUNDPROGOperation
- WSDL Binding: FUNDPROGHTTPSoapBinding

==> WSBind File Display and Change Utility ending
```

**Example 4:**

The control statements below display values in some WSBind files in a specified directory tree. The input controls statements plus the output are shown below.

**Input:**

```
#-> List WSBind files that start with 'f', end with 't'
FileNameFilter=f*t
ShowAllInDirectoryTree=/u/team10
```

(Note that you can also select WSBind files using a regular expression. If you want to specify a regular expression, use the `FileNameExpression=` command.)

**Output:**

```
==> WSBind File Display and Change Utility starting (V1.0.3)
==> Collecting statements. Statements will be scanned twice. If there are any
detected problems on the first pass, no actions will be taken.
==> -> stmt 1, comment => #-> List WSBind files that start with 'f' end with 't'
==> -> stmt 2, =====> FileNameFilter=f*t
==> -> stmt 3, =====> ShowAllInDirectoryTree=/u/team10

==> ==> Starting pass 1 (a syntax check).
==> ==> Starting pass 2 (taking actions).

==> -> statement 2 => FileNameFilter=f*t
==> File filter "f*t" interpreted to regular expression "f.*t"

==> -> statement 3 => ShowAllInDirectoryTree=/u/team10

Details for /u/team10/cicslab/wsbind/fundProgOut.wsbind
- Creation Timestamp: 20071024 12:47
- XML Conversion: Interpretive XML Conversion
  - Mapping Level: 2.0
  - Minimum Runtime Level: 2.0
- Requester
  - Endpoint URI: http://zserveros.dfw.ibm.com:9010/cicslab/fundProg
- WSDL File Name: C:\workspaces\cicsws\CICSRequestor\fundProgOut.wsdl
- Operation: FUNDPROGOperation
- WSDL Binding: FUNDPROGHTTPSoapBinding

Details for /u/team10/cicslab/wsbind/fundProgOutTest.wsbind
- Creation Timestamp: 20071024 12:47
- XML Conversion: Interpretive XML Conversion
  - Mapping Level: 2.0
  - Minimum Runtime Level: 2.0
- Requester
  - Endpoint URI: http://somehost2:12111/test/someService
- WSDL File Name: C:\workspaces\cicsws\CICSRequestor\fundProgOut.wsdl
- Operation: FUNDPROGOperation
- WSDL Binding: FUNDPROGHTTPSoapBinding

==> Number of WSBind files encountered during SHOWALLINDIRECTORYTREE command: 31
==> Number of WSBind files listed during SHOWALLINDIRECTORYTREE command: 2
==> Number of directories scanned during SHOWALLINDIRECTORYTREE command: 28

==> WSBind File Display and Change Utility ending
```



## Section 5: Installation

The artifacts supplied in this SupportPac are as follows:

- **dscutil.jar** – Jar file containing the BatchChangeWSBindFileContents Java program
- **UTWSB.SH** – a script file invoked by the PROC. The script file sets up the environment and invokes the Java program
- **DDWUTWSB.jcl** – a PROC that invokes the UTWSB script
- **UTWSBJCL.jcl** – sample JCL to invoke the DDWUTWSB PROC, with control statement input
- Version.txt – a file containing the version number of this SupportPac

### dscutil.jar

- Transfer dscutil.jar (**in binary mode**) to the same directory where you place UTWSB.SH. We transferred it to /u/team10/WSBindUtil.

### UTWSB.SH

- Transfer UTWSB.SH (**in text mode**) to an accessible location on your z/OS (same directory as the dscutil.jar). We transferred it to /u/team10/WSBindUtil on our system.
- Change the permission on the UTWSB.SH file to 755 so that it can be executed. The UNIX command to change permission on this file is 'chmod 755 /u/team10/WSBindUtil/UTWSB.SH'
- If your Java was not installed in a standard location, you may have to edit this file and change the export PATH statement to point at your Java installation
- Edit this file and change the CICS\_HOME environment variable to point to your CICS region's USS files. In our situation this was /usr/lpp/cicsts/cicsts42
- Edit this file and change the DSCUTIL\_JAR\_LOCATION environment variable to point to the directory where you transferred this file. In our environment this was /u/team10/WSBindUtil

### DDWUTWSB.jcl

- Transfer this JCL (**in text mode**) to a PROC LIB. Or, you can copy this PROC to a JCL library and change the JCL in UTWSBJCL to point to this JCL library with a JCLLIB statement. We used the JCLLIB statement approach in our environment.

### UTWSBJCL.jcl

- Transfer this JCL (**in text mode**) to a JCL library.
- Edit this JCL so that it contains a valid JOB card
- Edit this JCL and correct (or remove) the JCLLIB card
- Edit this JCL and in the line that says UTWSB=, specify the USS directory where you placed the UTWSB.SH script file
- This JCL file contains some sample control statements
- Change the control statements as appropriate
- Submit the JOB

## Section 6: Using the CS04 WSBIND File Utility

After the artifacts in this SupportPac are installed as per the previous section, just change the control statements as appropriate in UTWSBJCL (control statements are documented in section 2 of this document) and submit the JOB. The messages produced by this utility will be in your JOB log.

If the Java WSBIND utility program does not get executed, revisit the directions in Section 3 of this document. Additionally, it may help to understand the relationship of the artifacts (described in the next few paragraphs).

The Java utility program is in the **dscutil.jar** file. This Jar file should have been transferred to z/OS USS file system in binary mode because the Jar file contains Java byte-code (the equivalent of a Java load module).

The **UTWSB.SH** script is a USS script that sets some environment variables and invokes the WSBIND utility program that is in the dscutil.jar file. The WSBIND utility Java program uses some Java classes that are supplied with CICS TS V3.2 and higher. This means that the environment variables in the UTWSB.SH file must point to the location where Java is installed on your z/OS, must point to the location where you placed the dscutil.jar file, and must point to the CICS-supplied Java class needed by this utility program.

The **DDWUTWSB PROC**

- takes your control statements and places them into a tmp file in USS
- invokes the UTWSB.SH script and passes it the location of the control statements
- copies the messages produced by the utility to your JOB log

The **UTWSBJCL** is JCL that contains your control statement input to the utility program and invokes the DDWUTWSB PROC. The UTWSBJCL JCL contains a variety of control statements. Uncomment the ones you want to use or add new ones as appropriate.

## Section 7: How the Program Supplied with CS04 Works

The program supplied with CS04 provides a user interface that is intended to be invoked via a batch JOB. A PROC and sample JCL are supplied to execute the program. The sample JCL contains some sample control statements.

The program first reads all the control statements and lists their contents, indicating a statement number by each statement. Error message produced in the program that contain a statement number refer to the printed statement number.

The program then takes two passes at the data. The program processes all statements during both passes, with the exception that during the first pass, the program doesn't list any WSBIND file characteristics and does not write any WSBIND files. The reason for taking two passes at the statements is to try to ensure that the program doesn't stop half way through the control statements, with some statements processed and some statements not processed. If the program successfully gets through the first pass, there is a high probability that the second pass will be successful.

Statements are processed in the order in which they occur. The 'action' statements are DISPLAY, HELP, OUTPUT, RECURSEALL, or SHOWALL. If the input doesn't contain at least one action statement, an error message will be produced at the end of the first pass, and the program will stop. DISPLAY, RECURSEALL, SHOWALL, FILENAMEFILTER, and FILENAMEEXPRESSION statements must be before any INPUT statements. If an INPUT statement occurs, but no OUTPUT statement, an error message will be produced. An OUTPUT statement cannot occur until after an INPUT statement. DIRECTORY and SUFFIX statements can occur anywhere.

RECURSEALL, SHOWALLINDIRECTORYTREE, and SHOWALLINDIRECTORYSTRUCTURE are aliases for each other (i.e. they do the same). SHOWALL and SHOWALLINDIRECTORY are aliases for each other (i.e. they do the same).

### Displaying all files in a directory or directory tree...

When displaying all files in a directory, the program uses the Java File object to get a list of files in the directory. The file names in the file list are tested to see if they end in the appropriate suffix. The 'default' suffix is .wsbind, but that can be changed with the SUFFIX command. For file names passing the suffix test, the program then compares the file names to the appropriate 'file name expression'. The default file name expression is '.', which is the Regular Expression to allow any number of any characters. The file name expression can be changed with the FILENAMEEXPRESSION command. For information on how to compose a regular expression, consult the Java documentation (note that Java's use of regular expressions is different from PERL).

The FILENAMEEXPRESSION is assumed to contain a regular expression. The FILENAMEFILTER command can also be used to establish a file filter by using the '\*' and '?' wildcards. The program converts the FILENAMEFILTER into a regular expression for you. A valid FILENAMEFILTER might be F\*T, which will work if you want to display WSBIND files like FUNDMAINT. An example of a regular expression would be [Ff].\*[Tt], which would select files like FundOut, FUNDMAINT, and fundmnt. The FILENAMEEXPRESSION and FILENAMEFILTER settings only apply when selecting files in a directory or in a directory tree.

When recursing through a directory tree looking for WSBind files, the rules in the previous paragraphs apply when selecting WSBind files to display. The only difference is that a simple recursion algorithm is used to traverse the directory tree starting at the specified directory.

The commands allow you to be creative. For example, if you forgot where you placed the fundMaint.wsbind file, you could set FILENAMEEXPRESSION (or FILENAMEFILTER) to fundMaint, and then tell the program to recurse through your directories looking for the file. However, you could also use the UNIX 'find' command for this function.

When a WSBind file is found in a specified directory or directory tree (using the SHOWALL or RECURSEALL statements, the CICS TS V3.2 (and higher) Java WSBind file support classes are used to get the WSBind file contents, which are then 'printed' to stdout.

### Displaying specific WSBind file characteristics...

The DISPLAY statement is used to display the characteristics of a specific WSBind file. You can either fully qualify the WSBind file (such as /u/myhome/fundMaint.wsbind), omit the suffix (in which case the assumed suffix will be used), or just specify the file name (in which case the default directory will be used. The assumed suffix is .wsbind, but you can establish a new assumed suffix with the SUFFIX command. You can use the DIRECTORY command to establish a default directory (the default directory start out being set to "" (none)). The following sets of control statements do the same thing....

```
DISPLAY=/u/myhome/wsbind/FundMaint1.wsbind
DISPLAY=/u/myhome/wsbind/FundMaint2.wsbind
```

Or

```
DISPLAY=/u/myhome/wsbind/FundMaint1
DISPLAY=/u/myhome/wsbind/FundMaint2
```

Or

```
DIRECTORY=/u/myhome/wsbind
DISPLAY=FundMaint1.wsbind
DISPLAY=FundMaint2
```

Or

```
FileNameExpression=FundMaint[12]
ShowAll=/u/myhome/wsbind
```

## Changing the Characteristics of a WSBInd file...

The INPUT command is used to read a WSBInd file into memory. You may fully qualify the WSBInd file name, or a default directory may be established using the DIRECTORY command, and you can specify a suffix other than .wsbind using the SUFFIX command. Before attempting to read the file (using the CICS TS V3.2 (and later) supplied WSBInd file Java support classes), the program checks to see if the file exists and if the file is readable. Should these not be true, a message is produced.

To change the characteristics of the in-memory version of the WSBInd file (the WSBInd file specified on the INPUT statement), use the ENDPOINT, PGMNAME, etc control statements. When these control statements are used, the program uses the corresponding method from the CICS TS V3.2 (and later) supplied WSBInd file Java support classes.

To write the in-memory version of the WSBInd file, use the OUTPUT control statement. When this statement is encountered, the program uses the CICS TS V3.2 (and later) supplied WSBInd file Java support classes to write the file. You may fully qualify the name of the WSBInd file, or a default directory may be established using the DIRECTORY command, and you can specify a suffix other than .wsbind using the SUFFIX command.

If you change some characteristic of a WSBInd file, but forget to write it out with an OUTPUT control statement, the program will produce an error message.

Some combinations of control statements are invalid for some types of WSBInd files (requester vs. provider). The restrictions of the CICS TS V3.2 (and later) supplied WSBInd file Java support classes apply. Although the CS04 program checks for most invalid combinations, if the CICS TS V3.2 (and later) supplied Java support classes throw an exception, the exception is passed back to the user in the form of a message.

Some control statements, for example REQUESTCHANNELNAME, are only valid in combination with WSBInd files at certain mapping levels. The utility will provide an error message for invalid combinations.

Some control statements can only be used with certain levels of the dfjwsdl.jar file supplied with CICS. If you have an older version of dfjwsdl.jar than what is needed to perform a specified control statement, the utility will alert you of the problem and stop. For example, if you use the CICS TS V3.2 GA level of dfjwsdl.jar, you cannot use the WSDLFILENAME control statement. If you attempt to use a control statement that is beyond the capabilities of the WSBInd file support classes supplied with the maintenance level of your CICS, this SupportPac will provide an appropriate error message.

## Section 8: References

The capabilities of the CICS TS V3.2 provided WSBInd file Java support classes are documented at  
<http://publib.boulder.ibm.com/infocenter/cicsts/v3r2/index.jsp?topic=/com.ibm.cics.ts.webservices.javadoc/dfhwac00.htm>

CICS TS V3.2 online documentation (the InfoCenter) can be accessed at  
<http://publib.boulder.ibm.com/infocenter/cicsts/v3r2/index.jsp>

The CICS TS V4.1 provided WSBInd file Java support classes are documented in the CICS TS V4.1 InfoCenter.

The CICS TS V4.2 provided WSBInd file Java support classes are documented in the CICS TS V4.2 InfoCenter.

There are several excellent Redbooks on CICS's Web services support at  
<http://www.ibm.com/redbooks>

## One Final Comment:

If the CS04 program doesn't at least print some statements, there is probably something wrong with your setup, but other than that, should you find a problem, have comments, or suggestions, please send me an e-mail (Dennis Weiand - [dweiand@us.ibm.com](mailto:dweiand@us.ibm.com)).