SYBASE®

Command Line Tools Guide

e-Biz Impact™

5.4.5

DOCUMENT ID: DC10092-01-0545-01

LAST REVISED: July 2005

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book

Audience

This book is for e-Biz Impact users who want to execute various e-Biz Impact functionality from the command line or to execute commands from the Global Console.

How to use this book

The following topics are discussed in this book:

- Chapter 1, “Command Line Utilities,” provides a general overview and instructions on using the Command and Control (CNC) request utility.
- Chapter 2, “Using Command and Control Requests,” provides commands for e-Biz Impact executables.

Related documents

e-Biz Impact documentation The following documents are available on the Sybase™ Getting Started CD in the e-Biz Impact 5.4.5 product container:

- The e-Biz Impact installation guide explains how to install the e-Biz Impact software.
- The e-Biz Impact release bulletin contains last-minute information not documented elsewhere.

e-Biz Impact online documentation The following e-Biz Impact documents are available in PDF and DynaText format on the e-Biz Impact 5.4.5 SyBooks CD:

- The *e-Biz Impact Application Guide* provides information about the different types of applications you create and use in an e-Biz Impact implementation.
- The *e-Biz Impact Authorization Guide* explains how to configure e-Biz Impact security.
- *e-Biz Impact Command Line Tools Guide* (this book) describes how to execute e-Biz Impact functionality from a command line.
- The *e-Biz Impact Configurator Guide* explains how to configure e-Biz Impact using the Configurator.

- The *e-Biz Impact Feature Guide* describes new features, documentation updates, and fixed bugs in this version of e-Biz Impact.
- The *e-Biz Impact Getting Started Guide* provides information to help you quickly become familiar with e-Biz Impact.
- The *Monitoring e-Biz Impact* explains how to use the Global Console, the Event Monitor, and alerts to monitor e-Biz Impact transactions and events. It also describes how e-Biz Impact uses the standard Simple Network Management Protocol (SNMP).
- *Java Support in e-Biz Impact* describes the Java support available in e-Biz Impact 5.4.5.
- The *e-Biz Impact MSG-IDE Guide* describes MSG-IDE terminology and explains basic concepts that are used to build Object Definition Language (ODL) applications.
- The *e-Biz Impact ODL Guide* provides a reference to Object Definition Language (ODL) functions and objects. ODL is a high-level programming language that lets the developer further customize programs created with the IDE tools.
- The *e-Biz Impact TRAN-IDE Guide* describes how to use the TRAN-IDE tool to build e-Biz Impact production objects, which define incoming data and the output transactions produced from that data.

Note The *e-Biz Impact ODL Application Guide* has been incorporated into the *e-Biz Impact ODL Guide*.

The *e-Biz Impact Alerts Guide*, the *e-Biz Impact SNMP Guide*, and the *e-Biz Impact Global Console Guide* have been combined into a new guide—*Monitoring e-Biz Impact*.

Adaptive Server Anywhere documentation The e-Biz Impact installation includes Adaptive Server® Anywhere, which is used to set up a Data Source Name (DSN) used with e-Biz Impact security and authorization. To reference Adaptive Server Anywhere documentation, go to the Sybase Product Manuals Web site at Product Manuals at http://www.sybase.com/support/manuals/, select SQL Anywhere Studio from the product drop-down list, and click Go.

Note the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.

Other sources of information

Use the Sybase Getting Started CD, the SyBooks CD, and the Sybase Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

  Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.
- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Sybase Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on product certifications**

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Select Products from the navigation bar on the left.
3. Select a product name from the product list and click Go.
4. Select the Certification Report filter, specify a time frame, and click Go.
5. Click a Certification Report title to display the report.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1 Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

2 Select EBFs/Maintenance. Enter user name and password information, if prompted (for existing Web accounts) or create a new account (a free service).

3 Select a product.

4 Specify a time frame and click Go.

5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

The syntax conventions used in this manual are:

Key	Definition
commands and methods	Command names, command option names, utility names, utility flags, Java methods/classes/packages, and other keywords are in lowercase Arial font.
variable	Italic font indicates: • Program variables, such as *myServer* • Parts of input text that must be substituted, for example: `Server.log` • File names
File | Save	Menu names and menu items are displayed in plain text. The vertical bar shows you how to navigate menu selections. For example, File | Save indicates "select Save from the File menu."

Key	Definition
`package 1`	Monospaced font indicates: • Information that you enter in a graphical user interface, at a command line, or as program text • Sample program fragments • Sample output fragments

Accessibility features

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

For information about how Sybase supports accessibility, see Sybase Accessibility at http://www.sybase.com/accessibility. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

Command Line Utilities

This chapter describes the command line utilities provided by e-Biz Impact and how to configure e-Biz Impact to run command line options.

Introduction

The command and control (CNC) request utility allows you to control the e-Biz Impact environment from a Windows command line or a UNIX terminal window. CNC can be used to control e-Biz Impact clusters, controllers, and SFMs.

CNC must be run on the system where the e-Biz Impact server resides. The CNC utility executables are located in *x:\Sybase\ImpactServer-5_4\bin* on Windows and in *~/Sybase/ImpactServer-5_4/bin* on UNIX, where "*x*" and "~" represent the location where the e-Biz Impact server is installed.

Configuring e-Biz Impact to use CNC requests

Some e-Biz Impact commands require that you set specific environmental variables. To automatically set the required variables, a set of command wrappers is installed with e-Biz Impact. A wrapper is a command script, requiring one or more input parameters, that sets environmental variables and executes a binary.

Note Unless otherwise stated, these wrappers replace all command shells and environmental requirements in earlier versions of the product.

You can modify wrapper information to enable execution from different e-Biz Impact instance directories. Wrappers also enable you to seamlessly update e-Biz Impact regardless of version or executable name changes.

Adding the wrapper scripts directory to the PATH variable

Before you execute a wrapper script, you must set the *PATH* environment variable for the wrapper script directory. If you execute a command in a terminal window (UNIX systems) or at a DOS command line (Windows) and do not have the *PATH* variable set correctly, the system cannot find the command unless you include the full path to the executable when you type the command.

The wrapper script executables are located in *x:\Sybase\ImpactServer-5_4\bin* on Windows and in *~/Sybase/ImpactServer-5_4/bin* on UNIX systems, (where "*x*" and "~" represent the location in which the e-Biz Impact server is installed).

❖ **Setting the PATH variable on Windows**

1 Select Start | Settings, then double-click Control Panel.

2 When the Control Panel window opens, double-click the System icon.

3 When the System Properties window opens, select the Advanced tab, then click Environment Variables.

4 In the System Variables section of the window, select the *PATH* variable and click Edit.

5 In the Edit System Variable dialog box, append the *ims* wrapper script location to the PATH statement. For example, if the e-Biz Impact server is installed in the root of drive D: on Windows, enter this at the end of the PATH statement in the Variable Value field:

```
;D:\Sybase\ImpactServer-5_4\bin
```

Note Separate the existing path statement and the new entry with a semicolon (;) with no space after the semi-colon.

6 Click OK to save your entry and close the Edit System Variable dialog box.

7 Click OK to close the Environment Variable window, then click OK to close the System Properties window.

8 Close the Control Panel window.

9 Restart your machine to implement the new *PATH* variable.

❖ **Setting the PATH variable on UNIX systems**

This procedure is for the C-shell user environment. If your UNIX system uses a different shell (for example, the Bourne shell or Korn shell), see your UNIX system documentation for instructions on setting the *PATH* variable.

Note To see which shell is the default on your system, type `echo $SHELL` in a terminal window.

You can set the *PATH* variable each time you open a terminal session and start the cluster, or you can set the *PATH* variable permanently for all terminal sessions by adding the ims wrapper script path to the *$HOME/.cshrc* file.

Note Generally, shell variables apply only to the current instance of the shell and are used to set short-term working conditions; environment variables have a farther reaching significance, and those set at login are valid for the duration of the session. By convention, environment variables are uppercase and shell variables are lower case names.

The *PATH* environment variable and path shell variable specify directories to search for commands and programs. Both variables always represent the same directory list, and altering either automatically causes the other to be changed.

1 To permanently add the wrapper script directory to your path:

a Open the *$HOME/.cshrc* file in a text editor.

b Add the following line to the *.cshrc* after the list of other commands:

```
set path = ($path /<install_directory>/Sybase/ImpactServer-5_4/bin)
```

where <*install_directory*> is the location where the e-Biz Impact version 5.4.5 server is installed.

c Save the *.cshrc* file and close the text editor.

d Log out of the system and log back in to establish the new path.

2 To add the wrapper script path to the end of your existing path for only the current session, open a terminal window and enter the following on one line:

```
set path = ($path <install_directory>/Sybase/ImpactServer-5_4/bin)
```

where <*install_directory*> is the location where the e-Biz Impact version server is installed.

Note If you did not use step 1 to permanently add the wrapper script location to your path, each time you open a terminal window to execute the cluster, you must first enter the command shown in step 2 to set the PATH variable for the wrapper script for that session.

Wrapper scripts

The following table lists the wrapper scripts installed with e-Biz Impact.

Note Wrapper scripts have a *.cmd* filename extension on Windows, and a *.sh* filename extension on UNIX systems. In addition the

Wrapper script	Syntax	Description
ims	ims <*command*>	Executes all e-Biz Impact-related commands. • <*command*> is the command type followed by parameters specific to that command. Command types are alertd, alertg, bidl, cluster, clusterdm, clustersvc, cmdlinedfc, cnc, deploy, harvest, nidl, sfmlog, testlib, and xmlpretty.

Wrapper script	Syntax	Description
ims.db	ims.db <*command*>	Executes all commands associated with the authorization database. • <*command*> is any command you use on the database.
ims.nn	ims.nn <*command*>	Executes all NN configuration-related commands. • <*command*> is the command type followed by any parameters specific to that command. Command types are Crypt, Getmsg, and Putmsg. Be aware of case sensitivity.
ims.nt	ims.nt <*command*>	Executes all e-Biz Impact-related commands on only Windows NT. • <*command*> is the command type followed by any parameters specific to that command. Command types are ims54alertd, ims54alertg, ims54bidl, ims54cluster, ims54clustersvc, ims54cmdlinedfc, ims54cnc, ims54deploy, ims54harvest, ims54nidl, ims54sfmlog, ims54snmpproxy, ims54testlib, and ims54xmlpretty.
ims.setsrdir	ims.setsrdir <*port*>	Creates a directory for an SNMP port. This command only needs to be run once. • <*port*> is the SNMP port that you want to use.
ims.setsrports **Note** This script has a *.cmd* filename extension for Windows, a *.sh* extension for the UNIX Bourne shell, and a *.csh* extension for the UNIX C shell.	ims.setsrports <*port1*> <*port2*>	Sets the ports to which SNMP broadcasts telemetry and traps. • <*port1*> is the telemetry port. • <*port2*> is the trap port.
ims.sr	ims.sr <*command*>	Executes any SNMP-related command. • <*command*> is an SNMP command followed by any parameters specific to that command.

ims script

The *ims* wrapper script is used with each major e-Biz Impact component, including the e-Biz Impact and Adaptive Server Anywhere executables.

Note To reference Adaptive Server Anywhere documentation, go to the Technical Library Product Manuals Web site at Product Manuals at http://www.sybase.com/support/manuals/ and select Adaptive Server Anywhere from the product drop-down list.

The *ims* script contains the core of the executable name and associated command line arguments. For example, the binary used to run an e-Biz Impact cluster is ims54cluster. The command line executable for Windows and UNIX is:

```
ims cluster -cluster.name myCluster
```

where *myCluster* is the name of the cluster you want to run. Do not enter the cluster's filename extension (*.xml*).

For Windows NT, you must specify the full executable name:

```
ims.nt ims54cluster -cluster.name myCluster
```

Note See the *e-Biz Impact Configurator Guide*, Chapter 5, "Deploying Files and Executing e-Biz Impact Clusters," for additional arguments to use when you start clusters.

ims.nn script

Some commands require the prefix NN, such as NNCrypt, NNPutMsg, and NNGetMsg. These files are used in conjunction with Open Transport-XML implementations. See the *e-Biz Impact Monitoring Guide* and the *e-Biz Impact Application Guide* for more information.

NNCrypt

Description: Encrypts a file with a private key. Use NNCrypt to encrypt or decrypt the *nnsyreg.dat* configuration file.

Syntax:

NNCrypt {-encrypt | -decrypt} -file *fileName*

Parameters:

- -encrypt -file *fileName* – encrypt the specified file.
- -decrypt -file *fileName* – decrypt the specified file.

Encrypting a file does not change its functionality. After a file has been encrypted, the configuration files can be used the same as a file that is not encrypted.

NNGetmsg

Description

Performs a get of messages for testing. NNGetmsg reads an inbound transport, retrieves one message at a time, and writes each message to the output file until the transport is empty.

Syntax

NNGetmsg -o *output filename*
-t *transport ID*
-x *context ID*
[-p]
[-v]
[-pFile *parameter filename*]
[-pGroup *parameter group*]
[-m *message ID*]
[-commitInterval {number > 0 | ALL}]
[-blockingTimeout {time in milliseconds | INFINITE}
[-delimiter *delimiter*]
[-messageCount *n*]

Parameters

- -o *output filename* – required. The output file to which message are written. The user must have write privileges in the directory in which NNGetmsg is executed.
- -t *transport ID* – required. The transport ID from which the message is read.
- -x *context ID* – optional. The Open Transport context ID to use when retrieving messages from a transport. The default is “output”.
- -p – optional. Writes message properties to the output file (*output filename)*. The default is to write message properties to the screen.
- -v – optional. Turns on verbose mode where NNGetmsg logs are written to the screen. The defaults is no logging.
- -pFile *parameter filename* – optional. The name of the file from which to read program arguments.
- -pGroup *parameter group* – optional. The parameter group location within the parameter file.
- -m *message ID* – optional. NNGetmsg retrieves the next message matching the specified ID from the transport. The *messageID* can be specified in ASCII-encoded hexidecimal code because message IDs typically contain unprintable characters. An error is returned if the transport does not support retrieving messages based on the *messageID*.

- -commitInterval {number > 0 | ALL} – optional. The number of messages that can be received from the transport as part of a single transaction. If the commitInterval is larger than the number of available messages, NNGetmsg sends all available messages within one transaction. If the number of messages does not divide evenly into the commitInterval, the final transaction will contain fewer messages than the commitInterval value specified. The value must be greater than zero. The value "ALL" commits all messages in a single transaction. If the value is not specified, the default is one (1). This parameter is ignored if the transport is non-transactional.
- -blockingTimeout {time in milliseconds | INFINITE} – optional. Specify a blocking timeout in milliseconds or specify "INFINITE". NNGetmsg blocks on an empty transport for the time you specify before ending. An error is returned if the transport does not support blocking.
- -delimiter *delimiter*– optional. The delimiter used to separate messages in the output file. The value can be any number of printable or non-printable characters. Only application information is written to the output file. Control information, such as message properties, message descriptor fields, message count, or message size, is not written to the output file. The assumption is that the output file is intended for reload to a transport at a later date. Output files generated by NNGetmsg with the delimiter property are readable by NNPutmsg.
- -messageCount *n* – enter the number of messages to receive.

Remarks

NNgetmsg expects the defined transport name to exist and to be enabled.

Examples

Use the following examples for reference.

- Example 1 – this example uses the default context "output" to put all messages from the OutQ transport to the file *outputfile.txt*.

```
NNGetmsg -o outputfile.txt -t OutQ -v
```

- Example 2 – this example uses the "rules" context to put all messages from the OutQ transport to the file *outputfile.txt*. The message properties are also sent to *outputfile.txt*.

```
NNGetmsg -o outputfile.txt -t OutQ -v -x rules -p
```

NNPutmsg

Description

Performs a put of messages for testing. NNPutmsg reads messages from a file and puts the messages on the specified transport as defined by the *nnsyreg.dat* file.

Syntax

NNPutmsg -t *transport ID*
[-i *input filename*]
[-a *application group*]
[-m *message type*]
[-x *context ID*]
[-r]
[-S *component name*]
[-C {MSG | SUB}]
[-d]
[-v]
[-pFile *parameter filename*]
[-pGroup *parameter group*]
[-delimiter *delimiter*]
[-persistent]
[-nonPersistent]
[-messageProperties {*property name=property value*,
property name=property value,
property name=property value}]
[-commitInterval {number > 0 | ALL}]
[-repeatCount *number > 0*]
[-chunkSize *number > 0*]

Parameters

- -t *transport ID* – required. The ID of the transport where the message is sent.
- -i *input filename* – optional. The input file from which NNPutmsg reads. The file must reside in the directory from which the process is run, or the fully qualified path must be provided. The default is an empty message.
- -a *application group* – optional. The application group to associate with the message.
- -m *message type* – optional. Sets the message type.
- -s *context ID* – optional. Specifies the Open Transport context ID to use when sending the message. The default context ID is used if this parameter is not specified.
- -r – optional. Reloads the message.
- -S *component name* – required if -r is set. Identifies which component name to reload. If the process cannot set this property, a failure occurs and the process terminates.
- -C {MSG | SUB} – required if -r is set. Specifies the message type of the component being reloaded.
- -d – optional. Shuts down the message.
- -v – optional. Turns on verbose mode where NNPutmsg logs are written to the screen. The default is no logging.

- -pFile *parameter filename* – optional. Specifies the name of the file from which to read the program arguments.
- -pGroup *parameter group* – optional. Specifies the parameter group location within the parameter file.
- -d *delimiter* – optional. The delimiter used to parse messages. This value can be a printable or non-printable characters.
- -persistent – optional. Sends the message in persistent mode.
- -nonPersistent – optional. Sends messages in non-persistent mode.
- -messageProperties – optional. Allows multiple properties to be set for messages prior to sending the message to the transport. Specify binary property values using ASCII-encoded hexidecimal code. This parameter allows you to set transport-specific message properties, such as *AppIdentityData*, filed in the WebSphere MQ message descriptor. Using this method requires accuracy for the property name.
- -commitInterval {number > 0 | ALL} – optional. The number of messages to send as a single transaction. The value must be greater than zero. The value "ALL" sends all messages in a single transaction. If you do not specify this value, it defaults to one (1). If commitInterval is larger than the number of available messages, NNPutmsg sends all available messages within one transaction. If the number of messages does not divide evenly into the commitInterval, the final transaction will contain fewer messages than the commitInterval value specified. This parameter is ignored if the transport is non-transactional.
- -repeatCount *number > 0* – optional. Sends each message the number of times specified here. This value must be greater than zero. If this value is not specified, the default is one (1).
- -chunkSize *number > 0* – optional. Parses messages into chunks of the size specified by this parameter. This value must be greater than zero.

Remarks

Parsing messages from a single file Messages can be parsed by using a delimiter or a chunk size, depending on the parameter value specified. Because the delimiter and *chunkSize* parameters are mutually exclusive, specify a value for only one of these parameters. If neither a *delimiter* nor *chunkSize* is specified, NNPutmsg assumes the file contains one message.

Examples of delimiters include:

```
-delimiter 0x32
-delimiter @@@@
```

A *chunkSize* parses the input file into messages of fixed size, measured by byte count. For example:

```
-chunkSize 10
```

This example parses an input file containing 100 bytes into 10 messages, each 10 bytes in size.

Sending persistent or non-persistent messages All messages in a given run, from the start until the stop of the executable, are sent in persistent or non-persistent mode, depending on which parameter is set when NNPutmsg is started. Use the following guidelines to determine whether to use persistent or non-persistent mode:

- Persistent mode writes all messages to a persistent store. This allows data to be recoverable in the case of a system failure. With this mode, the time required by NNPutmsg to put each message is increased.
- Non-persistent mode stores messages in memory during processing. If there is a system failure, data is not recoverable.

  If a persistence mode is not specified, NNPutmsg defaults to the behavior specified by the transport. Some transports support only persistent messages, others support only non-persistent messages.

Examples

Use the following examples for reference.

- Example 1 – this example uses default context to put a message from *inputfile.txt* onto RulesIn transport with application group and message type properties set. The message contains all data from *inputfile.txt* until the end of the file.

```
NNPutmsg -i inputfile.txt
    -a TestApp
    -m TestFmt
    -t RulesIn
    -v
```

- Example 2 – This example uses rules context to put a message from *inputfile.txt* onto RulesIn transport with application group and message type properties set. Before message evaluation, a reload occurs for the subscription TestSub.

```
NNPutmsg -i inputfile.txt
    -a TestApp
    -m TestFmt
    -t RulesIn
    -v
    -x rules
```

```
    -r
    -C SUB
    -S TestSub
```

- Example 3 – this example uses rules context to put a message from *inputfile.txt* onto InQ transport with application group and message type properties set. The reload and component type signals a reload of the rule set before evaluating the message.

```
NNPutmsg -i inputfile.txt
    -a TestApp
    -m TestFmt
    -t InQ
    -v
    -x rules
    -r
    -C SUB
```

- Example 4 – this example uses rules context to put a message from *inputfile.txt* onto InQ transport with application group and message type properties set. The shutdown property signals a shutdown after message evaluation.

```
NNPutmsg -i inputfile.txt
    -a TestApp
    -m TestFmt
    -t InQ
    -v
    -x rules
    -r
    -d
```

Note See *Monitoring e-Biz Impact* and the *Open Transport 2.6 Configuration Guide* on the e-Biz Impact SyBooks CD that comes with the product, and the New Era of Networks Adapter for SAP R/3 3.9 User's Guide on the Sybase Product Manuals Web site for more information about using NN-type commands.

---

ims.db script

Adaptive Server Anywhere database components have a different prefix than e-Biz Impact, are installed to a different location than the e-Biz Impact executables, and use the *ims.db* wrapper.

Windows

Complete the following steps to configure and run the e-Biz Impact authorization database. The cluster name for this example is Cluster1.

1 Copy *impact.db* (default installation *C:\Sybase\ImpactServer-5_4\bin*) to *C:\Sybase\ImpactServer-5_4\Cluster1*.

2 Create a DSN for the e-Biz Impact client and server. See the *e-Biz Impact Authorization Guide* for instructions.

3 To configure the database to run as a service, enter this command on one line:

```
ims.db dbsvc -as -i -t network -s Manual
     -w Cluster1_db_service C:\Sybase\ImpactServer-5_4\asa\dbsrv8
     -n Cluster1_db
     -x "tcpip(PORT=NNNN)"
     C:\Sybase\ImpactServer-5_4\db\Cluster1\impact.db
```

Table 1-1: db parameters

Parameter	Value/description
-s	Manual or automatic. Specify Automatic to automatically start the Adaptive Server Anywhere database at machine startup.
-w	Identifies the name of the service appended to Adaptive Server Anywhere to form the service name displayed in the Windows Services control panel. In this example, "Adaptive Server Anywhere - Cluster1_db_service" would display.
-n	The logical name used by Adaptive Server Anywhere for this instance of the e-Biz Impact database—impact.db. In this example, Cluster1_db is used.
-x	Identifies port information. Replace *NNNN* with the TCP/IP port on which the database accepts connection requests. This port must be unique and unused by other applications on the localhost machine. The final value for this flag is the absolute path to the database file.

4 To start and stop the service, use the Windows Services control panel (Start | Settings | Control Panel | Administrative Tools | Services).

UNIX

Complete the following steps to configure and run the Impact database on UNIX. For this example, the cluster name is Cluster1 and the installation directory is */working/Sybase/ImpactServer-5_4/*.

1 Copy *impact.db* from */working/Sybase/ImpactServer-5_4/bin* to */working/Sybase/ImpactServer-5_4/clusters/Cluster1*.

2 Create a DSN for the e-Biz Impact client and server. See the *e-Biz Impact Authorization Guide* for instructions.

3 To start the database as a daemon, run this command on one line:

```
ims.db dbsrv8 -ud -m -n Cluster1_db -x "tcpip(PORT=NNNNN)"
     /working/Sybase/ImpactServer-5_4/clusters/Cluster1/impact.db
```

where:

Table 1-2: db parameters

Parameter	Description/Value
-ud	Prompts the database to run as a daemon. If you omit this parameter, the database remains on the command line.
-n	The logical name used by Adaptive Server Anywhere for this instance of the e-Biz Impact database—impact.db. In this example, Cluster1_db is used.
-x	Identifies port information. Replace *NNNN* with the TCP/IP port on which the database accepts connection requests. Note that this port must be unique and unused by other applications on the machine. The final value for this flag is the absolute path to the database file.

4 To shut down the database from the command line, enter `q` at the command prompt.

5 To shut down the database, if running as a daemon, send an interrupt signal (SIGINT) to the dbsrv8 process. Enter the following (as the user who started the process):

```
kill -2 pid
```

SNMP

To run any of the SNMP commands distributed with e-Biz Impact, use the *ims.sr* wrapper script provided in the *\Sybase\Impact-5_4\bin* directory. This ensures that the environment is set correctly. To run commands, pass the command as an argument to the script. For example, to run the SNMP daemon, use the ims.sr snmpdm -tcpany command. To run the daemon in debug mode, run the ims.sr snmpdm -d -tcpany command. Running the daemon in debug mode is useful for debugging setup issues.

Wrappers and scripts

The following table lists and describes the SNMP scripts.

Script	Description
ims.sr	A general wrapper script for SNMP commands.

Script	Description
ims.setsrdir	Sets the SNMP directory.
ims.setsrports.sh	Sets port environment variables using UNIX Bourne shell.
ims.setsrports.csh	Sets port environment variables using UNIX C shell.

Starting SNMP

Windows

To install the daemon as a service, enter:

```
ims.sr snmpdm -install -tcpany
```

To start a Windows service, do one of the following:

- Enter the command:

  ```
  ims.sr snmpdm -start
  ```

- From the Windows Services control panel (Start | Settings | Control Panel | Administrative Tools | Services) window, start the SNMP EMANATE Master Agent service.

UNIX

To start the daemon, enter the following on the command line:

```
ims.sr snmpdm -tcpany
```

To start the daemon in the foreground, specify the -d option on the command line:

```
ims.sr snmpdm -d -tcpany
```

Working with the Microsoft agent

To configure SNMP to run in conjunction with the Microsoft agent, complete these steps:

1 Stop the Microsoft SNMP service using the Windows Services control panel (Start | Settings | Control Panel | Administrative Tools | Services).

2 Go to *C:\WINNT\system32\drivers\etc*. Open the *services* file in a text editor and change this line:

```
snmp            161/udp            #SNMP
```

to this:

```
snmp            8161/udp           #SNMP
```

3 Save the file and close the text editor.

4 Use Start | Settings | Control Panel | Administrative Tools | Services to restart the Microsoft service.

Stopping SNMP

Windows

To stop a Windows service do one of the following:

- Open a command line window and enter this command at the prompt:

```
ims.sr snmpdm -stop
```

- Select Start | Settings | Control Panel | Administrative Tools | Services. When the Services window opens, right-click the SNMP EMANATE Master Agent service and select Stop.

UNIX

Find the SNMP process ID using the ps command, then use the kill command with the SIGKILL (9) signal.

For example:

```
kill -KILL <pid>
```

or

```
kill -9 <pid>
```

Where *pid* is the process ID that you identified from the ps command.

Other SNMP commands

To set up the standard port, enter:

```
ims.setsrdir 161
```

When installing a service, you must include the *-tcpany* option on the command line. For example, ims.sr snmpdm -install -tcpany.

Using alternate SNMP ports

Set the SNMP-related environment variables before you run the SNMP daemon, snmpdm.

- If you use the Bourne shell, use the *ims.setsrports.sh* script to set the port environment variables—*SR_SNMP_TEST_PORT* and *SR_TRAP_TEST_PORT*. The script takes the two ports values as command line arguments. For example, to use 5161 for the SNMP port and 5162 for the trap port, enter:

```
. ims.setsrports.sh 5161 5162
```

- If you use the C shell, copy the *ims.setsrports.csh* script to another name, for example a name that goes with the port values, then modify the values in the script values to the port values that you want to use.

Note For detailed instructions on using SNMP, see the *e-Biz Impact Monitoring Guide*.

Using cluster commands

The cluster commands described in this section use executables such as ims54cluster and ims54clustersvc. However, when you configure the *PATH* variable to locate the ims wrapper script, you only need to enter the portion of the command that follows "ims54". See "Adding the wrapper scripts directory to the PATH variable" on page 2 for instructions on setting the *PATH* variable.

cluster

Description: Runs a cluster.

Syntax:

ims -cluster.name *cluster_name*

Parameters: *cluster_name* – the name of the cluster to run.

clusterdm

Runs a specific cluster as a UNIX daemon. The command line syntax and arguments for clusterdm is identical to the cluster command.

clustersvc

Description: Runs or controls a specific cluster as a Windows service.

Syntax:

ims clustersvc -install *ServiceName* -home *dir* -file *config*

-setup *ServiceName* -home *dir* -file *config*

-remove *ServiceName*

-start *ServiceName*

-stop *ServiceName*

-restart *ServiceName*

Parameters

- -install *ServiceName* -home *dir* -file *config* – installs the cluster as a Windows service, where *ServiceName* is the name to give the cluster service (the default name is "ims54clustersvc"), *dir* is the name to give the home directory, and *config* is the cluster's configuration file.
- -setup *ServiceName* -home *dir* -file *config* – alters the home directory and configuration file for an existing service.
- -remove *ServiceName* – removes the service.
- -start *ServiceName* – starts the service.
- -stop *ServiceName* – stops the service.
- -restart *ServiceName* – restarts the service.

On Windows, you can also use Start | Settings | Control Panel | Administrative Tools | Services to start or stop the service after the service is installed and configured.

Using command line distributed function calls

The command line-to-DFC utility enables you to send DFC calls from within a shell script or from other types of applications that are not designed to use the normal DFC routing function commands.

You must create the DFC functions using MSG-IDE, and the DFCs must contain the strings you try to pass when you use the command line-to-DFC utility. Once you create the DFCs, they are saved to the AIM application's configuration file and are available for you to call from the command line.

Warning! When you define the function, the argument must have an [in] attribute.

To execute the utility, you can set the parameters in a configuration file or pass the options as command line arguments.

Note To run these command use the ims wrapper scripts, see "Adding the wrapper scripts directory to the PATH variable" on page 2.

You must also run this program from a directory where you have write permission since this program creates an *xlog* file in the directory from which you execute it.

Syntax

ims cmdlinedfc -cluster.name *clusterName*
-cluster.server *clusterServer*
-domain.type *domainType*
-domain.name *domainName*
-env.snmp.port *snmpPort*
-env.trap.port *trapPort*
-file *fileName*
-dfc.function *dfcFunctionName*
-dfc.flavor *dfcFlavor*
-dfc.args[*n*]

Required parameters

The following parameters are required in a command-line DFC call:

- -mgr *clusterName* – name of the cluster to receive the DFC call.
- -server *clusterServer* – the name of the machine on which the cluster is running. The default is "localhost" if not specified.
- -type *domainType* – type of the domain under which the cluster is running. The default value is "Impact."
- -name *domainName* – the name of the domain under which the cluster is running. The default value is "Impact."
- -function *dfcFunctionName* – name of the DFC function to call.
- -flavor *dfcFlavor* – flavor of the DFC to call. The default value is "0."
- -dfc.args *n* – list of string arguments to the DFC function. For example:

  ```
  -dfc.args arg1, arg2, arg3
  ```

  When arguments are provided on the command line, the *-dfc.args* parameter and its arguments must be the last arguments on the command line.

Optional parameters

These parameters are optional:

- -file *fileName* – the name of a configuration file that contains all or some command line arguments. Commands issued at the command line take precedence over the same commands in this file.
- -snmp.port *snmpPort* – the SNMP port to which the cluster publishes telemetries. The default value of is 161.
- -trap.port *trapPort* – the SNMP trap port to which the cluster publish alerts. The default value is 162.

Note The *cluster.name*, *domain.type* and *domain.name* arguments allow e-Biz Impact to identify the cluster external port using SNMP.

Log file

When the CNC utility is executed, the *dfc.xlog* log file is created. The log file contains debug, success, and error messages that occur during execution.

Return values

The return values are:

- 0 – the command is executed successfully. However, this does not mean that the command itself works, just that the command has executed properly. Check the actual command return value to verify that the command not only executed, but also did what it was supposed to do.
- 1 – the command failed to execute due to a DFC error.
- 2 – the command failed to execute due to an invalid configuration; for example, bad or missing arguments.
- 3 – the command failed to execute due to an SNMP error.
- 4 – the command failed to execute due to a failed connection to the e-Biz Impact server.
- 5 – the command failed to execute due to a failed login to the e-Biz Impact server.
- 100 – the command failed to execute due to an unknown error.

DFC argument files

If you create a script file, which can have a *.cfg* extension or a text file extension, contains information on where to find the cluster and the DFC commands that you want to call for that cluster. Commands issued at the command line take precedence over values listed in the configuration file. The file name is user-defined; for example, *<cluster_name>dfc.cfg*.

To execute DFC using a configuration file, use this syntax:

ims cmdlinedfc -file *filename*

where *filename* is the name of the configuration file that contains the arguments, values and the DFC to call.

Then you create a configuration file, the portion of the argument that precedes the period is the name of the section. The portion of the argument after the period is intended and below the appropriate section. For example, the argument -domain.type Impact would be entered like this in the file:

```
cluster
     name=clusterName
     server=localhost

domain
     name=Impact
     type=Impact

dfc
     function=dfcFunctionName
     flavor=1
     args=arg1 arg2 arg3
```

To invoke this command you enter:

```
ims cmdlinedfc -file mydfc.cfg
```

Alert commands

See *Monitoring e-Biz Impact* for information about alert commands.

BIDL utility

The BIDL utility is used to create and use custom C\C++ functions, in library form, callable within ODL projects. For example, to bridge C\C++-based interfaces with ODL. The utility generates code used in the creation of the shared library to handle the hand shake between ODL and your custom code.

The ODL engine loads a new library, generated based on your custom C\C++ function definition file, that correctly maps ODL calls to your custom C\C++ functions.

Usage:

```
ims.bidl <function definition file>
         <template file>
         <output file name>
```

Where <*function definition file*> is the file generated by the bidl code generator, based on your custom C function library; <template file> is the template file required to generate the code (provided with the e-Biz Impact 5.4 install); and <output file name> is the final library (*.*dll*) to be compiled.

See the *e-Biz Impact ODL Guide*, Chapter 8, “Using Shared Libraries,” for more information on generating libraries using the BIDL utility.

NIDL utility

The NIDL utility is used to process an Interface Definition Language (IDL) file and produce templates for custom plug-in applications. For example:

```
ims.nidl -a/-s idlfile.idl -d path
```

Where *-a* and *-s* indicate the application type (acquire or server), *-d* is optional, and indicates the path to the NIDL template file (template files are listed in the *e-Biz Impact Application Guide*).

sfmlog

The sfmlog utility extracts information from SFM log files. See the *e-Biz Impact TRAN-IDE Guide* for details.

xmlpretty

This command inserts line breaks into an XML file so that it can be viewed using a regular text editor rather than a XML-specific editor.

```
xmlpretty xml filename
```

where *xml filename* is the name of the XML file in which to insert line breaks.

CHAPTER 2

Using Command and Control Requests

This chapter describes the command and control (CNC) requests that can be issued from the command line or through the Global Console.

Introduction

A CNC request has three parameters:

- Subject – the subject executing the command.
- Command name – the action to execute.
- Object – (optional) the object on which the command is to be executed.

Only clusters, controllers, and SFM objects can accept CNC requests. The command name must be present in the list of commands for the appropriate object type.

A CNC command is either a CNC request (see “Using CNC requests” on page 33) or a CNC action (see“Using CNC actions” on page 41).

A CNC action is a set of one or more CNC requests issued to perform a specific task. For example, three CNC requests are necessary to resubmit unprocessable transaction.

An e-Biz Impact server that uses security is connected to a security database that contains all possible CNC requests that can be performed. Only authorized users can execute CNC requests. For more information on setting up user authorizations, see the *e-Biz Impact Authorization Guide*.

Executing CNC requests with the command line utility

To execute CNC requests, a command line utility is provided with the e-Biz Impact server binaries. To execute the utility, you must set the following options in either a configuration file or pass the options as command line arguments.

Note To run these command use the ims wrapper scripts, see "Adding the wrapper scripts directory to the PATH variable" on page 2.

Syntax

ims cnc -cluster.name *cluster_name*
-cluster.server *server_name*
-domain.type *domain_type*
-domain.name *domain_name*
-file *file_name*
-cnc.user *user_name*
-cnc.password *password*
-cnc.subject *subject*
-cnc.type {mgr | ctlr | sfm}
-cnc.object *object*
-cnc.command *command*
-cnc.arg*n*
-env.snmp.port *snmp_port*

Required parameters

The following parameters are required in a CNC request:

- -usr *user_name* – the user name of the person invoking the command. If you are using e-Biz Impact security, this user must have a role assigned that authorizes them to use this command. See the *e-Biz Impact Authorization Guide* for more information.
- -pwd *password* – required. The password for the user name that was entered.

- -sub *subject* – the subject executing the action.
- -type *{mgr | ctlr | sfm}* – the type of object for which the command is being issued—the cluster (*mgr*), the controller (*ctlr*), or the SFM (*sfm*).
- -obj *object* – the object for which the request is being issued.
- -cmd *command* – the command being issued.

Optional parameters

These arguments are optional in a CNC request:

- -mgr *cluster_name* – name of the cluster for which the command is being executed.
- -server *server_name* – the name of the machine on which the cluster is running. The default is "localhost" if not specified.
- -type *domain_type* – the cluster's domain type. The default is "Impact" if not specified.
- -name *domain_name* – the cluster's domain name. The default is "Impact" if not specified.
- -file *file_name* – the name of a configuration file that contains all or some command line arguments. Commands issued at the command line take precedence over the same commands in this file.
- -snmp.port *snmp_port* – optional. The SNMP port only if the cluster publishes telemetry on a custom port. When using custom SNMP ports, you must specify this argument. If you do not specify this argument, the CNC command fails. See *Monitoring e-Biz Impact* for more information about using SNMP.
- -args *n* – an optional list of string arguments to the CNC function. For example:

  ```
  -cnc.args arg1, arg2, arg3
  ```

  When arguments are provided on the command line, the argument values must be the last arguments on the command line.

Note The *cluster.name*, *domain.type* and *domain.name* arguments allow e-Biz Impact to identify the cluster external port using SNMP.

Log file

When the CNC utility is executed, the *cnc.xlog* log file is created. The log file contains debug, success, and error messages that occur during execution.

Return values

The return values are:

- 0 – the command is executed successfully. However, this does not mean that the command itself works, just that the command has executed properly. Check the actual command return value to verify that the command not only executed, but also did what it was supposed to do.

  For example, if getTransactionData is issued with a serial number that does not exist, the command itself could be executed properly, and the return value would be `0`. However, since the serial number is invalid, the command itself would return -900.
- 1 – the command failed to execute due to a CNC error.
- 2 – the command failed to execute due to an invalid configuration; for example, bad or missing arguments.
- 3 – the command failed to execute due to an SNMP error.
- 4 – the command failed to execute due to a failed connection to the e-Biz Impact server.
- 5 – the command failed to execute due to a failed login to the e-Biz Impact server.
- 100 – the command failed to execute due to an unknown error.

Examples

This example of a CNC request entered at the command line disables a cluster. The command would be entered on one line:

```
ims cnc -mgr cluster1 -type mgr -sub cluster1
     -obj cluster1 -cmd disable -usr system
     -pwd manager
```

This CNC example enables a controller:

```
ims cnc -mgr cluster1 -type ctlr -sub controller1
     -obj controller1 -cmd enable -usr system
     -pwd manager
```

Note The *-type* and *-sub* arguments must match. To enable or disable applications other than SFMs, use the application name for the subject.

CNC argument files

You can create a file that contains all or some command line arguments. Commands issued at the command line take precedence over values listed in the configuration file. The file name is user-defined; for example, *<cluster_name>cnc.cfg*.

To execute CNC using a configuration file, use this syntax:

ims cnc -file *filename*

where *filename* is the name of the configuration file that contains the CNC argument values and commands.

When you create a configuration file, the portion of the argument that precedes the period is the name of the section. The portion of the argument after the period is intended and below the appropriate section. For example, the argument -domain.type Impact would be entered like this in the file:

```
domain
     type=Impact
```

This example shows a configuration file named *mycnc.cfg* that enables an SFM controller named SFM1 running on the MultiProcess cluster.

```
cluster
     name=MultiProcess
     server=localhost

domain
     type=Impact
     name=MultiProcess

cnc
     user=system
     password=manager
     type=ctlr
     subject=SFM_Controller
     object=SFM1
     command=enable
```

To invoke this command you enter:

```
ims cnc -file mycnc.cfg
```

Examples

The examples in this section assume that any executed commands are authorized for the user JOHN with the password PWD. SFM1 is running on e-Biz Impact CLUSTER1 on machine HOST1 under domain type PROD1 with the domain name DOMAIN1 publishing SNMP data through custom port 4545.

The examples illustrate how to execute CNC requests on the following objects using a configuration file named *myCncFile.cfg*.

- HOST1 is a computer name (can be resolved through the DSN)
- PROD1 is the domain type of the server
- DOMAIN1 is the domain name of the server
- CLUSTER1 is an object of type cluster
- CONT1 is an object of type controller
- SFM1 is an object of type SFM
- AIM1 is a child application of CONT1
- DEST1 is a child destination of SFM1

Note When using custom SNMP ports, you must specify the "env.snmp.port" argument. If you do not specify this argument in the file (for example, `env snmp.port 6163`), the CNC command fails. See *Monitoring e-Biz Impact* for more information about using SNMP.

SFM refuse mode

This example requests that the SFM object SFM1 refuses to dispatch transactions to all of its destinations.

Syntax

```
ims cnc –file myCncFile.cfg
```

Configuration file

myCncFile.cfg contains:

```
cluster
     name=CLUSTER1
     server=HOST1

cnc
     user=JOHN
     password=PWD
     subject=SFM1
     type=sfm
     command=refuseTransactions

domain
     type=PROD1
     name=DOMAIN1

env
```

```
snmp.port=4545
```

Disabling a controller

This example requests cluster object CLUSTER1 to disable its child controller CONT1.

Syntax

ims cnc –file myCncFile.cfg

Configuration file

myCncFile.cfg contains:

```
cluster
     name=CLUSTER1
     server=HOST1

cnc
     user=JOHN
     password=PWD
     subject=CLUSTER1
     type=mgr
     command=disable
     object=CONT1

domain
     type=PROD1
     name=DOMAIN1

env
     snmp.port=4545
```

Pausing destinations by transaction serial number

This example requests the SFM object SFM1 to pause all destinations to which the transaction is dispatched.

Syntax

ims cnc –file myCncFile.cfg

Configuration file

myCncFile.cfg contains:

```
cluster
     name=CLUSTER1
     server=HOST1

cnc
     user=JOHN
     password=PWD
     subject=SFM1
     type=sfm
     command=pauseDestsBySerial
```

```
        arg[1]=1221234

    domain
        type=PROD1
        name=DOMAIN1

    env
        snmp.port=4545
```

Disabling applications (including the SFM)

This example requests the cluster object, CLUSTER1, to disable its child controller object, CTL1, and to disable its child application, SFM1.

Syntax

ims cnc -file myCncFile.cfg

Configuration file

myCncFile.cfg contains:

```
cluster
     name=CLUSTER1
     server=HOST1

cnc
     user=JOHN
     password=PWD
     subject=CTL1
     type=ctlr
     command=disable
     object=SFM1

domain
     type=PROD1
     name=DOMAIN1

env
     snmp.port=4545
```

Executing CNC requests from the Global Console

Some CNC requests and actions can be executed from the Global Console. When an agent is connected and displays a running e-Biz Impact server, the server can have more that one cluster. Each cluster has its own authorization database when you use e-Biz Impact security. See the *e-Biz Impact Authorization Guide* for more information on enabling cluster object security.

Note Before you can monitor cluster activity, the cluster must be running. See the *e-Biz Impact Configurator Guide*, Chapter 5, "Deploying Files and Executing e-Biz Impact Clusters," for instructions.

❖ **Issuing CNC commands on cluster objects**

1 On Windows, select Start | Programs | Sybase | e-Biz Impact 5.4 | Global Console.

2 Log in to the cluster and provide a valid user name and password, select an object in the tree view, right-click, select All Tasks, then select the operation you want to perform.

3 When you select the command, the Global Console sends a set of CNC requests to the bus. The bus delegates those commands to the subjects for which they were issued. The subject then executes the commands one by one.

4 All commands usually execute a predefined set of CNC requests that automatically set the subject, object (if any), command name, and command parameters based on the context from which the commands were invoked.

Note Executing the action indicates if the bus received it successfully or not. Because the communication to the bus uses an asynchronous dialog, the return from the CNC action does not guarantee that the action was successful. The bus can be configured to sends alerts. Alerts can be set up to be raised when major events happen or by responding to CNC actions. Information on configuring alerts is available in *Monitoring e-Biz Impact*.

Using CNC requests

The tables in this section list all possible CNC requests grouped by object type. The command description gives a quick summary of what the CNC request does. The ID field identifies the request that is referenced later in the documentation.

These commands are authorized. If the authorization fails the command returns 3300. In authorization mode, all commands are authorized. If a command is not routable (specifically, the object or subject name provided can not be found) it returns 3310.

Table 2-1: Cluster commands

ID	Command	Description
M3	disable	Request a cluster object to stop a child controller. • Subject is cluster name • Object is controller name • No arguments • Returns 1for success, 3310 if command is not routable
M4	enable	Request a cluster object to start a child controller. • Subject is cluster name • Object is controller name • No arguments • Returns 1for success, 3310 if command is not routable
M5	reload	Reloads a cluster object's configuration data from its configuration file and restarts all of its controllers and application. • Subject is cluster name • No object • No arguments • Returns 1for success, 3310 if command is not routable
M6	shutdown	Request a cluster object to stop all of its SFM objects, controller objects, and applications managed by the controllers. • Subject is cluster name • No object • No arguments • Returns 1for success, 3310 if command is not routable
M7	getStatus	Request a cluster object to indicate the status of a child controller. • Subject is cluster name. • Object is child controller name or blank. • No arguments. • Returns 1 for enabled; 2 for disabled; 3 for enabling; 4 for disabling. When no object is specified, the cluster returns its own status: 1 for enabled; 2 for disabled; 3 for enabling; 4 for disabling; 5 for reloading.

Note 3310 is a generic error, which means the error is not an unauthorized or unrouteable action.

Table 2-2: Controller commands

ID	Command	Description
C3	disable	Request a controller object to disable a child application. • Subject is controller name • Object is child application name • No arguments • Returns 1for success, 3310 if command is not routable
C4	enable	Request a controller object to enable a child application. • Subject is controller name • Object is child application name • No arguments • Returns 1 for success, 3310 if command is not routable
C5	getStatus	Request a controller object to indicate the status of a child application. • Subject is controller name. • Object is child application name • No arguments • Returns 1 for enabled; 2 for disabled; 3 for enabling; 4 for disabling

Table 2-3: SFM commands

ID	Command	Description
S1	acceptTransactions	Request an SFM object to resume dispatching transactions to all its destinations. • Subject is the SFM's name • No object • No arguments • Returns 700 for success, -700 for failure

ID	Command	Description
S4	cancelTransactionBySerial	Request an SFM object to cancel a pending transaction identified by a serial number and a destination name. • Subject is SFM name • No object • argument[1] = serial number or • Subject is SFM name • No object • Argument[1] = destination name • Argument[2] = *serial serial* or *serial1 serial2* for range • Returns 400 for success, -400 for failure
S5	cancelUnprocessableTransaction	Request an SFM object to cancel an unprocessable transaction identified by a serial number. • Subject is SFM name • No object • Argument[1] = *serialnumber serialnumber* • Returns 400 for success, -400 for failure
S8	deleteUnroutableTransaction	Request an SFM object to delete an unrouteable transaction identified by a serial number. • Subject is SFM name • No object • Argument[1] = serial number • Returns 1100 for success, -1100 for failure
S9	getStatus	Request an SFM object to retrieve the status of a source or destination. • Subject is SFM name • Object is source or destination name • Argument[1] = 0 for source, 1 for destination • Returns (source) -1 for error, 0 for inactive (alive but not currently processing data), 1 for active, 2 for dynamic, 3 for pinging, and 4 for failed (ping DFC). • Returns (destination) -1 for error, 0 for inactive, 1for active, 2 for paused, 3 for stuck, 4 for invalid, 5 for retrying, 6 for failed, and 7 for sleeping.

ID	Command	Description
S10	getTransactionData	Request an SFM object to retrieve transaction data identified by a serial number. • Subject is SFM name • No object • Argument[1] = log type (0 for unprocessable, 1 for unrouteable, 2 for pending (including unprocessable), 3 for cancelled) • Argument[2]= serial number • Returns 900 for success, -900 for failure • Add the data in the argument list
S11	getTransactionList	Request an SFM object to build a list transactions, starting at a row number and for a specified number of rows. Specify a destination name to list transactions pending in a destination queue. • Subject is SFM name • No object • Argument[1] = log type (0 for unprocessable, 1 for unrouteable, 2 for pending transactions (including unprocessable), 3 for cancelled) • Argument[2] = start row index • Argument[3] - number of rows to return • Argument[4] = (optional) destination name • Returns 900 for success, -900 for failure • Add the list in the argument list
S12	pauseAllDest	Request an SFM object to stop dispatching transactions to all of its destinations. • Subject is SFM name • No object • No arguments • Returns 200 for success, -200 for failure
S13	pauseDest	Request an SFM object to stop dispatching transactions to a specified destination. • Subject is SFM name • Object is destination name • Arguments[1] = destination name • Returns 200 for success, -200 for failure

ID	Command	Description
S14	pauseDestsbySerial	Request an SFM object to stop dispatching transactions to the destination for which the given serial number was destined. • Subject is SFM name • No object • Argument[1] = serial number • Returns 200 for success, -200 for failure
S15	refuseTransactions	Request an SFM object to stop dispatching transactions to all destinations. • Subject is SFM name • No object • No arguments • Returns 600 for success, -600 for failure
S16	repairInvalidDest **Note** The security databases in versions of e-Biz Impact prior to 5.4.5 contained the replaceInvalid command, which has been replaced by repairInvalidDest.	Request an SFM object to substitute an invalid destination with a valid destination. • Subject is SFM name • No object • Argument[1] = invalid destination type (AIM, SFM, and so on) • Argument[2] = invalid destination flavor • Argument[3] = new destination type (AIM, SFM, and so on) • Argument[4] = new destination flavor • Returns 900 for success, -900 for failure
S17	resendTransactionBySerial	Request an SFM object to resend completed transactions specified by a serial number to either a single destination or all destinations. • Subject is SFM name • No object • Argument[1] = *serial serial* or *serial2 serial2* for range or • Subject is SFM name • No object • Argument[1] = destination name • Argument[2] = *serial serial* or *serial2 serial2* for range • Returns 500 for success, -500 for failure

ID	Command	Description
S18	resubmitTransaction	Request an SFM object to resubmit a transaction specified by a serial number. • Subject is SFM name • No object • Argument[1] = log type (0 for unprocessable, 1 for unrouteable, 2 for pending transactions (include unprocessable), 3 for cancelled) • Argument[2] = transaction data • Argument[3] = serial number • Argument[4] = production object name (unprocessable only) • Argument[5] = destination flavor (unprocessable only) • Argument[6] = destination type (unprocessable only, for example, AIM, SFM, and so on) • Returns 1000 for success, -1000 for failure
S19	resumeAllDest	Request an SFM object to resume dispatching transactions to all destinations. • Subject is SFM name • No object • No arguments • Returns 300 for success, -300 for failure
S20	resumeDest	Request an SFM object to resume dispatching transactions to a specific destination. • Subject is SFM name • Object is destination name • Arguement[1] = destination name • Returns 300 for success, -300 for failure
S21	resumeDestBySerial	Request an SFM object resume dispatching transactions to all destinations for which the specified serial number was destined. • Subject is SFM name • No object • Arguement[1] = serial number • returns 300 for success, -300 for failure

ID	Command	Description
S22	skipTransactionBySerial	Request an SFM object to skip one or more transactions to either a single destination or all destinations. • Subject is SFM name • No object • Argument[1] = *serial serial* or *serial1 serial2* for range or • Subject is SFM name • No object • Argument[1] = destination name • Argument[2] = *serial serial* or *serial1 serial2* for range • Returns 400 for success, -400 for failure
S23	skipUnprocessableTransaction	Request an SFM object to skip an unprocessable transaction specified by a serial number. • Subject is SFM name • No object • Argument[1] = *serial serial* • Returns 400 for success, -400 for failure
S24	uncancelTransactionBySerial	Request an SFM object to uncancel one or several transactions to either a single destination or all destinations. • Subject is SFM name • No object • Argument[1] = *serial serial* or *serial1 serial2* for range or • Subject is SFM name • No object • Argument[1] = destination name • Argument[2] = *serial serial* or *serial1 serial2* for range • Returns 1200 for success, -1200 for failure
S25	reprocessUnprocessableTransaction	Request an SFM object to reprocess an unprocessable transaction identified by serial number. • Subject is SFM name • No object • Argument[1] = *serial serial* • Returns 400 for success, -400 for failure

ID	Command	Description
S26	SendToSFM	Send a new transaction to an SFM. • Subject is SFM name • No object • Argument[1] = source name used as reference • Argument[2] = route • Argument[3] = transaction data • Argument[4] = route type • Returns 1 for success • Add the new transaction serial number in the argument list • Add the return value of the route call in the argument list

Note The ims54cnc utility logs returned arguments to the *cnc.xlog* file. Therefore, do not use the command line tool for commands that accept an argument in the return; for example, getTransactionData (S9) and getTransactionList (S10), or any other unprocessable, unrouteable, and repair actions.

Using CNC actions

Complex actions, such as resubmitting a transaction to a specific route, usually include one or more CNC requests combined in a CNC action.

Resending transactions

To resend a transaction to all destinations, use:

- pauseDestsBySerial
- resendTransactionBySerial
- resumeDestsBySerial

To resend a transaction to a specific destination, use:

- pauseDest
- resendTransactionBySerial

- resumeDest

Skipping transactions

To skip a transaction for all destinations, use:

- pauseDestsBySerial
- skipTransactionBySerial
- resumeDestsBySerial

To skip a transaction for a specific destination, use:

- pauseDest
- skipTransactionBySerial
- resumeDest

Cancelling transactions

To cancel a transaction for all destinations, use:

- pauseDestsBySerial
- cancelTransactionBySerial
- resumeDestsBySerial

To cancel a transaction for a specific destination, use:

- pauseDest
- cancelTransactionBySerial
- resumeDest

Uncancelling and repairing cancelled transactions

To access the cancelled transactions SFM view, use:

- getTransactionList

To repair a cancelled transaction, use:

- pauseDestsBySerial

- uncancelTransactionBySerial
- resumeDestsBySerial

To uncancel a transaction for all destinations, use:

- pauseDestBySerial
- uncancelTransactionBySerial
- resumeDestBySerial

To uncancel a transaction for a specific destination, use:

- pauseDest
- uncancelTransactionBySerial
- resumeDest

Working with unprocessable transactions

To access the unprocessable transactions SFM view, use:

- getTransactionList

To skip an unprocessable transaction, use:

- pauseDestsBySerial
- skipUnprocessableTransaction
- resumeDestsBySerial

To cancel an unprocessable transaction, use:

- pauseDestBySerial
- cancelUnprocessableTransaction
- resumeDestBySerial

To resubmit an unprocessable transaction, use:

- getTransactionData
- resubmitTransaction

Working with unrouteable transactions

To access the unrouteable transactions SFM view, use:

- getTransactionList

To delete an unrouteable transaction, use:

- deleteUnrouteableTransaction

To resubmit an unrouteable transaction, use:

- getTransactionData
- resubmitTransaction

Index

I

N

P

R

S

T

U

W

X