Beta Systems Architecture

Installation and System Guide

Version 7 Release 1

March 22, 2021

Document number: BSA-V7R1-INS-EN-20210322

Beta Systems DCI Software AG
Alt-Moabit 90d
D-10559 Berlin

www.betasystems-dci.com

Support Contact Information

support@betasystems.com

Telephone Germany:
0800-BETASYS (or 0800-2382797)

Telephone International:
+49 (0)6321 499 15 108

Contents

Introduction 8
Overview 9
BSA components 11
What is new in BSA 14
  Level 1771-00 - New features and functions 14

License check handling 18
License file 19
How licenses are checked 22
Monitoring license-related messages 24
Definition of terms used in a license file 25

Installation 28
Hardware and software requirements 29
Pre-installation checklist 30
Overview of installation steps 34
Step 1: Preparing the installation datasets 36
Step 2: Running the installation REXX 40
Step 3: Submitting installation batch jobs 42
Step 4: Specifying authorized functions to z/OS 48
  Activating authorized functions (Beta SVC) in z/OS 49
Defining the subsystem ID to z/OS and initializing the subsystem 51
  IEFSSNxx entry for defining/initializing subsystem at IPL 52
  SETSSI command for defining/initializing subsystem dynamically 53
  BST01ARI: Initializing subsystem and security environment 54
  Security definitions required for subsystem initialization 57

Customization 59
Beta parameter library (BETA.PARMLIB) 60
  Examples of started task (SFF) and batch job (RFF) procedures 61
  Specifying parameters 63
  Selection of parameter members 65
  Parmlib member syntax 66
BSA LST parameters and how to change them 68
  LST parameters in B01LSTxx 69
  LST parameters in BnnLSTxx 71
  Specifying different routing codes 74
  Suppressing messages 76

Suppressing dumps for SFF/RFF and SFF storage regions 77
Static system symbol support 79

RACF security 81
User resource class $BETA 82
Security exit global parameters 84
Refreshing the RACF security user table 85

Multi-CPU with the Open Communication Facility (OCF) 86
Introduction 87
Overview of OCF functions and parameters 88
OCF environment 89
Configuring OCF for a multi-CPU environment 91
Customization 92
VTAM resource definitions 93
OCF network resource definitions 94
LST parameters for OCF 95
Calculating the values for RU sizes 98
Security in a host-to-host multi-CPU environment 100
OCF operator console commands 101
Examples of OCF configurations 102
Example 1: Host-to-host - 4 STCs and 4 OCF nodes (recommended) 103
Example 2: Host-to-host - 4 STCS and 2 OCF nodes 105
Establishing OCF communication via X-System Router (Beta 02) 107
Example 1: X-System Router Beta 02 with slave system 108
Example 2: X-System Router Beta 02 without a slave system 109
Parameters for OCF via TCP/IP 110

Multi-CPU using BSA XCF in a sysplex 116
Introduction 117
Which multi-CPU configuration types are possible? 118
Prerequisites and requirements for BSA XCF communication 119
Initializing subsystems in the sysplex 120
Communication with an SSID not in the sysplex 122
Example: Communication between sysplex and non-sysplex 123
Working with the sysplex 124
Displaying XCF members, subsystem and STC info 126
Configuration examples 128

Creating an address space with BSA XCF Global Connect 130
Introducing BSA XCF Global Connect 131
System overview of BSA Global XCF 133

Activating BSA XCF Global Connect 134

BSA TCP/IP server 136

Introduction.. 137

Server functions and features 138

BSA TCP/IP server ports.. 139

Compression and encryption for the BSA TCP/IP server 141

BSA TCP/IP server logon exit (B02UXSIN) 142

BSA TCP/IP server as a standalone STC (Beta 02)............................. 144

BSA TCP/IP server in a Beta product STC 146

BSA TCP/IP server as an RFF job for EDF 147

SFF client groups for a TCP/IP port 148

Operator commands for the TCP/IP server.. 150

LST parameters for the BSA TCP/IP server 154

Securing TCP/IP connections with AT-TLS 163

Configuring Policy Agent as a started task in z/OS........................... 164

Defining the security authorization for Policy Agent.......................... 165

Defining the Policy Agent configuration files 166

Configuring AT-TLS... 167

Defining the AT-TLS policy rules... 169

Creating and configuring digital certificates in RACF 172

Securing a connection between BSA TCP/IP server and client partner 173

BSA Communication Integrator (BSA CI) 174

Introducing the BSA CI... 175

System overview 177

BSA CI functions and features 180

Installing and customizing BSA CI 182

Creating JCL for the BSA CI started task.. 184

BSA CI ports... 185

Defining a service port.. 186

Dynamic server control and automatic retry.. 187

Available runtime modes 188

Client authentication and authorization 191

Defining SSL authentication security 192

Defining a security environment and certificates............................... 193

Creating and customizing a member for key ring and other parameters 200

Creating and customizing a member for port and other parameters 203

LST parameters for BSA CI.. 204

Compression and encryption for the BSA CI 210

zIIP Facility (ZIF) 211
Introducing ZIF 212
LST parameters for ZIF 213

Automatic restart management (ARM) support 214
Enabling ARM support 215
Registering started tasks with ARM 216
Activating ARM 217
ARM policy 218

Databases and database batch utilities 219
Introduction 220
Global LST parameters for BQL 221
Dynamic database extension 224
Merging concatenated VSAM datasets 228
Sharing databases with the BQL share option 232
Serialization 235
Activating IBM's Global Resource Serialization (GRS) 236
Synchronization file (SYNC) 237
BnnCLSYN: Cleaning the SYNC file 239
Notes on model spool file definitions 241
JCL for BSA database batch utilities 242
BST05CMD: Updating the database definition file 244
BST05UPF: Updating file information in the database definition file 246
Example: Activating database mirroring 248
Example: Deactivating database mirroring 252
BST05LAU: Locking and unlocking a BQL database 254
Loading data into a BQL database (BST05DBL) 255
Unloading data from a BQL database (BST05DBL) 257
Formatting a database component (BST05FOR) 258
Building or rebuilding keys for a table (BST05DBL) 260
Deleting the contents of a table, key, or file (BST05DBL) 261
BST05ANA: Database analysis and repair utility 262
BST08OCP: Copying/Checking/Repairing/Restoring archive files 270
Using concurrent copy to back up a running BQL database 272

Base Archive Facility (BAF) 275
LST parameters for BAF 276

Base Output Facility (BOF) 279
Introduction 280
DD statements in the BOF started task procedure 281
Running BOF as a separate started task 282

BOF default LST parameters for VTAM/APPC 284

Diagnostic reports and traces... 286

Information for Beta Systems support 287

BSA trace parameters 288

BSA trace media 290

Allocating a trace medium 291

Releasing a trace medium... 293

Activating/Deactivating BSA traces 294

Diagnostic reports 295

Beta SMP/E package information 298

Third-party software.. 300

Index 301

Introduction

In this chapter

Topic **Page**

Overview ... 9
BSA components ... 11
What is new in BSA ... 14
  Level 1771-00 - New features and functions ... 14

Overview

This manual

This manual describes how to install Beta Systems Architecture (BSA) and the BSA components. The installation of the Beta product functions is described in the appropriate product manual. However, always keep in mind that BSA has to be installed first.

What is BSA?

BSA is the underlying architecture of all Beta products. It provides the basic system functions used by these products and product enhancement facilities, such as management of the product databases.

Because the BSA functions are identical for all Beta products, we recommend that you use one set of BSA libraries for all the Beta products you install.

Installation medium

Beta Systems distributes its software via download from the Beta Systems website. BSA is included in the product download.

You will need to register when visiting this page for the first time:

`https://customer.betasystems.com/`

You will also need an appropriate license file to run licensed products.

Installation procedure

The installation procedure performs three basic functions:

- It sets up the SMP/E environment for all products.
- It installs all BSA functions.
- It calls the Beta product installation procedure(s) and passes the information necessary for product installation (for example, the names of the SMP/E datasets). This step is optional.

You can install the BSA components only, without extensions like _beta vaf, _beta caf or _beta iaf. The first Beta product can also be installed later. This is possible because all the information entered when you install BSA is stored and retrieved whenever the installation procedure is called again.

Important: The installation REXX of a product includes only the BSA components that are used by this product. Use the setup program of the product that includes all shared components that you need in your environment. See "Pre-installation checklist" on page 30 for more details.

Installing BSA and Beta products

The BSA components have to be installed before you can operate a Beta product. The BSA functions can either be installed separately, or at the same time as a Beta product is installed. Whenever you install a new product, you should first determine whether another Beta product has been installed previously. If this is the case, we recommend that you use the same SMP/E and BSA environment.

A consolidated installation process facilitates the installation of BSA components and Beta products. All installation processes are started by means of the Beta installation REXX that is included in the installation datasets.

SMP/E environment

When you install BSA, an SMP/E environment is set up for BSA and Beta products. The installation descriptions in this manual and in the Beta product manuals assume that you use only one common SMP/E environment for all Beta products and the BSA functions. This SMP/E environment should be used exclusively by Beta products.

Advantages of using the same environment

The installation descriptions in the BSA and product *Installation and System Guide* assume that you use one common SMP/E environment for the BSA functions and all Beta products. They also assume that you use only one set of BSA libraries for all Beta products.

Using just one environment makes the maintenance of BSA components much easier to handle. Because the BSA functions are essential for correct Beta product operation, it is strongly recommended that these functions are always kept at the highest available maintenance level. If you have only one environment, you can be sure that all the Beta products installed in it are operating on an equally high level of quality.

BSA components

Overview

This section provides an overview of the BSA components.

SFF - Subsystem Function Facility

The Subsystem Function Facility (SFF) is the heart of the BSA platform and provides the operating environment in which most application functions will execute. SFF services are written with performance and availability in mind. SFF services include:

- Task dispatching
- Storage control
- Program control
- Locking services
- Dynamic allocation
- Cross-address-space communication
- Cross-system coupling facility (XCF)
- Message formatting
- Trace facilities
- Dump formatting
- Console communication
- Asynchronous dispatching of work requests

BSF - Base System Facilities

Base System Facilities (BSF) is a set of services used in common by BSA and product components. BSF services include:

- z/OS dataset access
- Data compression/decompression
- Communication with SFF-based systems
- Date conversion and formatting
- ISPF interface
- Program linkage
- Programming
- Security call routing
- Switch management
- Syntax checking
- Trace service

BOF - Base Output Facility

The Base Output Facility (BOF) provides a set of services for managing printing requests to 3270-type printers (or emulations). SNA LU types 0, 1, 3 and 6.2 protocols are supported. BOF supports printing to JES and printing to file, and manages the printing to VTAM printers initiated by _beta doc|z.

BAF - Base Archive Facility

The Base Archive Facility (BAF) provides a set of services for writing to archives, reading from archives, and administering archived data.

DMF and BQL database facilities

The Data Management Facility (DMF) and Beta Query Language (BQL) consist of a set of database access services providing keyed, sequential, or indexed table database access services in an SFF operating environment. All information is stored and retrieved in VSAM ESDS datasets providing an optimum in performance, reliability and maintainability.

DMF and BQL also provide facilities for coordinating database activities between their own applications and z/OS dataset services and facilities. These include services for locking resources, software caching, and dataset backup services in a shared-DASD multi-CPU environment.

OCF - Open Communication Facility

The Open Communication Facility (OCF) provides peer-to-peer communication services between programs written for an SFF operating environment. It also provides functions running in other environments or other SFF operating environments.

OCF uses the Advanced Program-to-Program Communication (APPC) services of z/OS to facilitate communications with these external functions. Alternatively, you can also use OCF via TCP/IP.

XCF – Cross Coupling Facility

BSA XCF communication uses the extended capabilities of the Cross Coupling Facility (XCF) component of the z/OS operating system. It enables different Beta products and CPUs to be accessed across LPAR boundaries without requiring the use of an additional task such as the OCF Cross System Router.

VDF - VTAM Dialog Facility

The VTAM Dialog Facility (VDF) is a set of services for managing the dialog between 3270-type display terminals running in a native VTAM (non-TSO) environment and applications running in an SFF operating environment. VDF enables users in non-TSO environments to access Beta products.

The add-ons _beta caf and _beta iaf enable access to Beta products from CICS and IMS operating environments. VDF is the basic component for the operation of the add-ons _beta vaf, _beta caf, and _beta iaf. These add-ons are included in the TES (Transaction Environment Support) license article.

CAF - _beta caf

_beta caf provides extensions to the VDF to allow for the display of VDF-formatted screens in the CICS operating environment.

Separate documentation is available for _beta caf.

IAF - _beta iaf

_beta iaf provides extensions to the VDF to allow for the display of VDF-formatted screens in the IMS operating environment.

Separate documentation is available for _beta iaf.

VAF - _beta vaf

_beta vaf provides extensions to VDF to allow VDF formatted screens to be displayed in a native VTAM environment.

Separate documentation is available for _beta vaf.

RPG - _beta report

_beta report is a powerful tool for generating reports in batch.

_beta report provides a wide range of predefined reports. You can also design your own reports to meet special requirements.

Separate documentation is available for _beta report.

BSM - BSA Service Manager

The BSA Service Manager enables dynamic control of the products in the running system. It displays the running activities and connections and facilitates the maintenance and support of functions. The functionality is integrated into an online application and can run within a product ISPF online application, or can be used as a separate ISPF application.

Separate documentation is available for the BSA Service Manager.

What is new in BSA

Level 1771-00 - New features and functions

Spool model check

Beta Systems products with spool files will output a warning message when the number of available model spool files is too low:

```
9546W SPOOL FILES:n SPOOL MODELS:n TOTAL CYLINDERS OF SPOOL MODELS:n
```

_beta log|z also creates an event (see "Standard events" in *_beta log|z Administrator Guide*).

The warning threshold is defined via the LST parameter BQL_SPOOLCHECK_MODEL (Allowed:1..9 Default:1).

The check is carried when the STC is started and when a new spool file is allocated.

IPv6 support

Support of IPv6 addresses has been added to all functions of the BSA TCP/IP server and the BSA Service Manager.

IPv6 support has also been added to the BSA service routines for the TCP/IP communication in order to enable Beta V7 products to use IPv6.

The changes that have been implemented ensure downward compatibility with IPv4.

Note: IPv6 support has **not** been added to OCF TCP/IP communication (BSA X-System Router) and the BSA Communication Integrator (BSA CI).

PORT LST parameter

The optional parameter <resolve> has been added to the PORT LST parameter of the TCP/IP server. The <resolve> parameter controls which numeric IP addresses are used for the bind following the resolution of a symbolic IP address.

Syntax

```
Bnn_TCPIP_PORT[_app] = port,ipa[<resolve>],tcpname[,[clnt],[mssid],[rssid],[keepalive]]
```

Parameter	Description	Opt./Req.
port	Listening port (max. 5 digits)	required
ipa[<*resolve*>]	Bind address (numeric or symbolic) Use the following notations if you want to specify an all-zeros address to bind to all available interfaces: 0.0.0.0 All available IPv4 addresses; accepts only connections via IPv4 protocol. IPA_ANY4 Same as **0.0.0.0** :: All available IPv4 and IPv6 addresses; accepts connections via IPv4 and IPv6 protocol. IPA_ANY6 Same as **::** Symbolic addresses (max. 255 characters) are resolved to numeric addresses (IPv6 and/or IPv4) via DNS. The resolution sequence is determined by the DNS server. By default, IPv6 addresses have precedence over IPv4 addresses. You can use the ***<resolve>*** parameter to control DNS resolution if ***ipa*** is a symbolic address: *ipa* The TCP/IP server uses all addresses from DNS resolution and binds to the first reachable IPv6 address and to the first reachable IPv4 address. This is the default, which is adequate in most situations. *ipa*<IPV6> The TCP/IP server uses all IPv6 addresses from DNS resolution and binds to the first reachable address. *ipa*<IPV4> The TCP/IP server uses all IPv4 addresses from DNS resolution and binds to the first reachable address. *ipa*<IPV6:ALL> The TCP/IP server uses all IPv6 addresses from DNS resolution and binds to all reachable addresses. *ipa*<IPV4:ALL> The TCP/IP server uses all IPv4 addresses from DNS resolution and binds to all reachable addresses. *ipa*<ALL> The TCP/IP server uses all IPv6 and IPv4 addresses from DNS resolution and binds to all reachable addresses. The ***<resolve>*** parameter has no effect if ***ipa*** is a numeric address.	required
tcpname	Name of the TCP/IP started task (TCP/IP stack) on the z/OS system (max. 8 characters)	required

Parameter	Description	Opt./Req.
other parameters	See the description of the PORT LST parameter in "LST parameters for the BSA TCP/IP server" on page 154 in *BSA Installation and System Guide*	optional

Example

DNS resolves TEST03.ZS.DE.EU.BETA.ADS to the following three addresses:

- 2001:8db:a:75:8f65:5b26:9720:4ab1
- 10.56.83.100
- 10.56.83.101

If the following parameter is coded in the active LST member:

```
B93_TCPIP_PORT_BWE = 64001,TEST03.ZS.DE.EU.BETA.ADS<IPV4>,TCPIP
```

The BSA TCP/IP server binds to...

- 10.56.83.100:64001

...if this is the first reachable IPv4 address. The BSA TCP/IP server cannot be reached at the other two addresses.

If the following parameter is coded in the active LST member:

```
B93_TCPIP_PORT_BWE = 64001,TEST03.ZS.DE.EU.BETA.ADS,TCPIP
```

The BSA TCP/IP server binds to:

- [2001:8db:a:75:8f65:5b26:9720:4ab1]:64001
- 10.56.83.100:64001

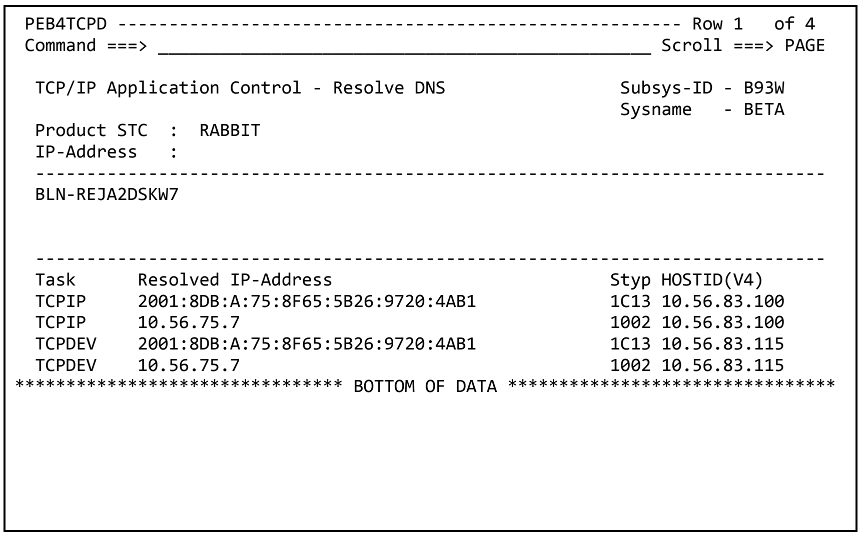
Code the following parameter in the active LST member if you want the BSA TCP/IP server to be reachable at all addresses:

```
B93_TCPIP_PORT_BWE = 64001,TEST03.ZS.DE.EU.BETA.ADS<ALL>,TCPIP
```

BSA Service Manager

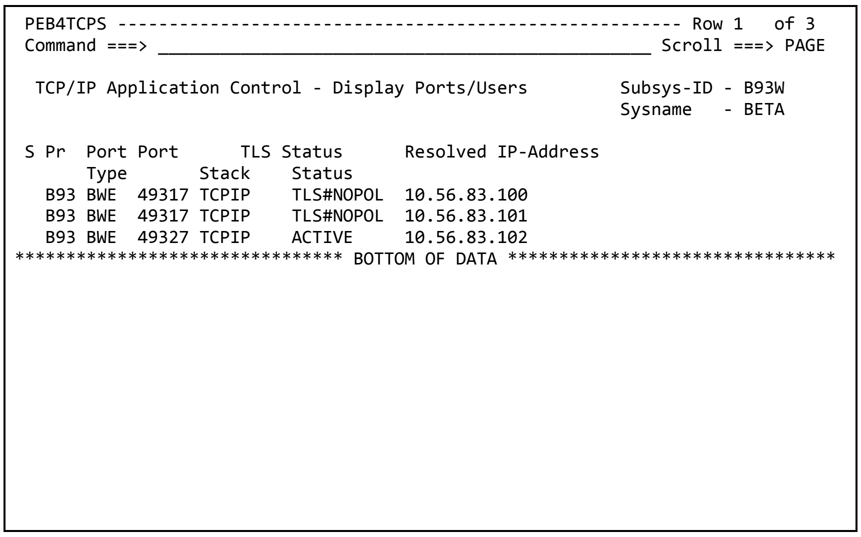
New functions have been added to the BSA Service Manager option **4.2.2** (TCP/IP connectivity) with respect to IPv6 support.

For example, you can use the BSA Service Manager to check the resolution of symbolic IP addresses:



```
 PEB4TCPD -------------------------------------------------------- Row 1   of 4
 Command ===> ________________________________________________ Scroll ===> PAGE

  TCP/IP Application Control - Resolve DNS              Subsys-ID - B93W
                                                        Sysname   - BETA
  Product STC  :  RABBIT
  IP-Address   :
  -----------------------------------------------------------------------------
  BLN-REJA2DSKW7


  -----------------------------------------------------------------------------
  Task      Resolved IP-Address                        Styp HOSTID(V4)
  TCPIP     2001:8DB:A:75:8F65:5B26:9720:4AB1          1C13 10.56.83.100
  TCPIP     10.56.75.7                                 1002 10.56.83.100
  TCPDEV    2001:8DB:A:75:8F65:5B26:9720:4AB1          1C13 10.56.83.115
  TCPDEV    10.56.75.7                                 1002 10.56.83.115
******************************** BOTTOM OF DATA ********************************
```

The BSA Service Manager can also be used to display TLS status information on addresses/ports:



```
 PEB4TCPS -------------------------------------------------------- Row 1   of 3
 Command ===> ________________________________________________ Scroll ===> PAGE

  TCP/IP Application Control - Display Ports/Users       Subsys-ID - B93W
                                                         Sysname   - BETA

 S Pr  Port Port      TLS Status      Resolved IP-Address
       Type       Stack    Status
   B93 BWE  49317 TCPIP    TLS#NOPOL  10.56.83.100
   B93 BWE  49317 TCPIP    TLS#NOPOL  10.56.83.101
   B93 BWE  49327 TCPIP    ACTIVE     10.56.83.102
******************************** BOTTOM OF DATA ********************************
```

For a complete description, see the *BSA Service Manager Manual*.

Earlier changes

For a list of earlier changes, see *BSA V6R1 Installation and System Guide*.

License check handling

In this chapter

Topic	Page
License file	19
How licenses are checked	22
Monitoring license-related messages	24
Definition of terms used in a license file	25

License file

Overview

The license file that you have received from Beta Systems contains the required license information for your licensed Beta Systems products, add-ons and applications.

It is not possible to run a Beta Systems product without an appropriate entry in a valid license file.

Obtaining a license file

Please contact Beta Systems order desk (OrderDesk@betasystems.com) if you need a new license file.

Order desk needs the following information to be able to generate a valid license file for you:

- Your name
- The name of your company
- The product(s) / add-on(s) to be included in the license file
- The CPU type *tttt-cccc*, i.e. the processor type (*tttt* - the last four digits of the serial number), and the number of configured processors (*cccc*). You can obtain this information from the startup message `xxx9151I CPU INFORMATION - TYPE...` of an installed Beta product or via the console command `D M=CPU`. For the latter, proceed as follows:
  1. Call SDSF.
  2. Enter the primary command `LOG`.
  3. Enter the console command `/D M=CPU` and send the output of this command to OrderDesk@betasystems.com.

```
IEE174I 10.50.45 DISPLAY M 510
PROCESSOR STATUS
ID  CPU                SERIAL
00  +                   01BCXCtttt
01  +                   01BCXCtttt
02  +A                  01BCXCtttt
03  +I                  01BCXCtttt

CPC ND = 00XXXX.S08.IBM.XX.00000XXXBCXC
CPC SI = XXXX.XXX.IBM.XX.000000000XXXBCXC
```

Other product-dependent information may be required depending on your contract.

License types

Three types of license are supported:

MIPS	MIPS (Million Instructions Per Second) licensing assumes that there is a direct relationship between the size of a computer and, for example, the number of jobs that can run on it and/or the number of lists it can produce. The MIPS size is calculated on the basis of the type of machine and number of processors. How the product reacts under certain circumstances is determined by the license policy.
Usage	Usage licensing is based on product-specific factors that enable its usage (load) to be calculated. These factors could include jobs per day, pages per month, number of records to be administrated, number of concurrent users, etc. They are collected by the product and monitored by a reporting procedure. The UBP policy determines product behavior and specifies the steps to be taken if licensed volume is exceeded.
Server	This type of license is associated with a single machine that is identified by its DNS name.

DCB attributes

The following DCB attributes are supported when you transfer your license file to a dataset on the z/OS host:

Dataset organization:	PO or PS
Record format:	FB, VB, U
Logical record length:	Minimum value is determined by the longest record in the license file (LRECL=512 recommended)

The license file has to be transferred in text mode, i.e. with ASCII to EBCDIC translation. Do not modify the license file in any other way, because this would invalidate it. Copy it to other LPARs if necessary.

License file name

To enable the Beta product or application to carry out a license check, it needs to know where the license information is stored. There are two ways of providing the product with the required reference to the license file:

- Via the LST parameter B*nn*_LICX_DSNAME=*datasetname* in the parmlib member used by the product (This is the recommended method; it supports dynamic update.)
- Via the DD statement B*nn*LICX in the JCL (This method does not support dynamic update.)

If both are specified, the DD statement has precedence over the LST parameter.

When is a new license needed?

Normally, a single license file contains all license information and is shared by the products.

Under the following circumstances you will need to obtain a new license file from Beta Systems order desk (OrderDesk@betasystems.com):

- when one of your existing licenses expires
- when a new CPU type is added to your installation or the number of configured CPUs increases
- when you want to be able to use additional products or new licensable add-ons
- when an update to a new product generation requires a different license key

When license information has changed

When you have received a new license file from Beta Systems to replace your old one, you have to make this updated license information available to the affected STCs.

For some changes (for example, later expiration date or different license type), this update can be carried out dynamically with the help of the BSA Service Manager (option **2.4** of the "Service Manager Selection Menu"; see the *BSA Service Manager Manual* for details). Dynamic update is only supported if the license file has been defined via the LST parameter B*nn*_LICX_DSNAME.

For other changes (for example, new CPU type), the STC(s) must be stopped and restarted.

See *Release Notes and Update Instructions*

Before installing or updating a product or add-on, please see the product *Release Notes and Update Instructions* to find out whether an updated license file is required.

How licenses are checked

Overview

Beta products and applications check for the presence of a valid license. Normally, a single license file contains all license information and is shared by the products.

License check time

Depending on the product or application you are using, the license check is carried out when the product is started, and afterwards cyclically at midnight (or another hour if you have coded the LST parameter B*nn*_LICX_CHECK_TIME=*hh*).

This applies to master and slave STCs and to jobs that are running as database masters (SIGNON=NO).

What happens when a license is found

When a license file is found for the Beta product that is starting, it will be checked. The following may occur:

- The license file contains a valid license. The product STC is started.
- There is a discrepancy in the license file. What happens next depends on the discrepancy and the product concerned:
  - If the license is a trial version (InstID=UNSPECIFIED, installation type=TRIAL) and the trial time has not yet expired, the product STC will be started with the appropriate message.
  - If the license is a trial version (InstID=UNSPECIFIED, installation type=TRIAL) and the trial time has already expired, the product STC will not be started and an error message will be generated.
  - If the license contains toleration parameters (depends on the license type), the product STC will be started with the appropriate message.
  - If the license file has been changed in any way, the product STC will not be started and an error message will be generated.
  - If the basic license is valid, but the add-on license for CAF/VAF/IAF is invalid, the online user will receive the appropriate message.
- If no license file is found, it will not be possible to start the product.

Termination date

"License will expire on *yyyy-mm-dd*" indicates the termination date of a license. *yyyy-mm-dd* is the last date when this license is valid. License checks carried out on a later date will detect that this license has expired. By default, the STC detects this at the very beginning of the following day (B*nn*_LICX_CHECK_TIME=00). If the product license is affected, the STC will normally stop and you will be unable to restart it with this license file.

Tolerance period: If the license is permanent and its policy is non-enforced, the STC will continue to run for another 42 days after the termination date.

Goodwill period: The goodwill period does not provide any extension beyond the termination date, but it enables you to run a product for up to 30 days in a different environment, for example, on a standby machine. Reaching the end of goodwill does not invalidate the license file. It can be used when the product is transferred back to the original environment, and it can later be used on the standby machine in case of another emergency.

Product add-ons

When the database master started task is checked, all the product add-ons anchored in the product are also checked.

Usage license check

If the type of license being checked is a usage license, the check will take place globally. However, the usage values themselves will be checked by each licensed product separately.

If you have a server license

If you have a server license, you must ensure that the product STCs (database servers) or users submitting the batch job for a MASTER RFF batch job either have their own OMVS segments or use the default OMVS segment. They will need READ access to facility class BPX.DEFAULT.USER.

Automatic default OMVS segments

To set up RACF so that it automatically uses default OMVS segments for users and groups that do not have their own OMVS segments in their USER or GROUP profiles, proceed as follows:

- Create a FACILITY class profile called BPX.DEFAULT.USER with universal access READ and APPLDATA(*defaultuser/defaultgroup*).
- Define a RACF user profile containing an OMVS segment for *defaultuser*.
- Define a RACF group profile containing an OMVS segment for *defaultgroup*.

Monitoring license-related messages

Overview

The product STC writes warning messages to notify you when licenses are about to expire. We strongly recommend that you monitor these messages with the help of a monitoring tool or an automatic operator.

Messages to be monitored

The following warning messages are written to the JESMSGLG during the warning or goodwill period (normally 30 days) each day at midnight (or another hour if specified via the LST parameter B*nn*_LICX_CHECK_TIME=*hh*):

```
9001W product LICENSE WILL EXPIRE IN n DAYS ...
9001W product LICENSE WILL EXPIRE ON yyyy-mm-dd ...
9003W addon LICENSE WILL EXPIRE IN n DAYS ...
```

The following error message should also be monitored, which is written when the STC is unable to start a particular addon:

```
9002E addon NOT ACTIVE - ADDON LICENSE HAS EXPIRED ...
```

When one of these messages occurs, please contact your Beta Systems sales representative or Beta Systems order desk (OrderDesk@betasystems.com) for a new license file. You can then activate the new license during a planned restart of the product or dynamically with the help of the BSA Service Manager option **2.4** (see "Updating license information" in *BSA Service Manager Manual*).

Other informational messages

Additionally, you can also monitor the STC log for these informational messages if you want to be informed about license expiration dates before the warning or goodwill period:

```
9001I Bnn LICENSE WILL EXPIRE ON yyyy-mm-dd
9002I addon LICENSE WILL EXPIRE ON yyyy-mm-dd
```

Definition of terms used in a license file

Overview

The following table lists and describes all the terms used for and in the license file.

Term	Description
instID	Installation ID. This is the unique identifier of the system (database) that has been installed. Unless **unspecified** is defined, the Installation ID is created by the product the first time it is started. instID is used as the unique search criterion for an installation in a license file.
product	This refers to the actual Beta product. It describes the installation defined by the Installation ID, and is the name of the schema that specifies which entries can be made for this product in the license file. Add-ons and enhancement facilities are declared as articles within the product.
start	The starting date of a license period in format yyyy-mm-dd. Before this date, the license is invalid.
termination	The end date of a license period in format yyyy-mm-dd. After this date, the license is invalid and will terminate after x days, depending on the license term and the license policy.
licenseTerm	This field shows the license termination type, either permanent or temporary. **permanent**: The product has been purchased. **temporary**: The product has been acquired for a specific period of time. This can be a one-off fee for a defined period of time, or the rental for the product.
licenseType	The license type indicates the conditions of contract and which units of measurement are to be applied. There are three license types (see "License file" on page 19): **MIPS** **Usage** **Server**
licensePolicy	The license policy defines how a product will behave under certain circumstances: **enforced:** This parameter specifies that when a product reaches a specific threshold value without being renewed, it will not allow any further functions, or that the system stops or cannot be started when a mainframe used with a MIPS license is upgraded. **tolerant:** Threshold values can be exceeded, but the deviations are logged. A mainframe with a MIPS license can be upgraded, but a warning will be issued stating the goodwill period.

Term	Description
installationType	The installation type specifies how the product can be used and includes the boundary conditions for this use: **Trial**: This type of installation enables a product or product component to be tested. It can be used when installing the product for the first time. **Test**: This is a fully-fledged product installation that is not being used for production, but for tests or integration tests instead, with a variety of other software. Its purpose would be to map the infrastructure of the productive system. **Production**: A fully-fledged production system. **Standby**: These are systems that are kept on standby for emergencies, or are only used for production in transitional situations.
cpuType	This value defines the processor type (*tttt* – the last four digits of the serial number) and the number of configured processors (*cc*).
serverName	The DNS name is used to check dedicated servers.
article name	Name of an add-on ("**Base**" indicates the base product).
period	The period of validity of an add-on (does not apply to Base). **value:** The starting point of the period of validity. **value2**: The end of the period of validity.
goodwill	The number of days the product can still be used if certain parameters in the license file are invalid, e.g. invalid cpuType or serverName. Once the goodwill period has elapsed, the product will stop, regardless of the license period.
warning	When a license is due to expire, a warning message will be issued from this point in time onward.

Example of a license file

```
<?xml version="1.0" encoding="UTF-8"?>
<lic:definition xmlns:lic="http://www.betasystems.com/schemas/licenseDefinition">
  <lic:body hash="12345678"  hashID="B-CODE-01"   >
    <lic:creation date="2005-11-07" time="13-14-30"/>
    <lic:customer name="BETA SYSTEMS"  id="12345678" >
      <lic:address street="Alt Moabit 90d"  city="D-10559 Berlin"  country="German"
           city2=""  name="BETA SYSTEMS" name2="" name3="" number="90d" region=""  zip="10559"/>
    </lic:customer>
    <lic:installation Product="B92 - MIPS" goodwill="30" instID="B92-ABCD-EFGH-IJKL"
                      installationType="Trial" licensePolicy="Enforced" licenseTerm="Permanent"
                      start="2005-11-07" termination="2008-11-07" warning="30">
      <lic:articles>
        <lic:article name="Base">
          <lic:param name="licenseType" value="MIPS"/>
          <lic:param name="cpuType" value="2084-0002"/>
        </lic:article>
        <lic:article name="OSY - Enterprise Agent">
          <lic:param name="license" value="YES"/>
          <lic:param name="period" value="2005-11-07" value2="2008-11-07"/>
        </lic:article>
        <lic:article name="VAF - VTAM Access">
          <lic:param name="license" value="YES"/>
          <lic:param name="period" value="2005-11-07" value2="2008-11-07"/>
        </lic:article>
        <lic:article name="XCF - XCF Support">
          <lic:param name="license" value="YES"/>
          <lic:param name="period" value="2005-11-07" value2="2008-11-07"/>
        </lic:article>
      </lic:articles>
    </lic:installation>
  </lic:body>
</lic:definition>
```

Note Any manipulation of the license file will invalidate it.

Installation

In this chapter

Topic **Page**

Hardware and software requirements 29
Pre-installation checklist 30
Overview of installation steps 34
Step 1: Preparing the installation datasets 36
Step 2: Running the installation REXX 40
Step 3: Submitting installation batch jobs 42
Step 4: Specifying authorized functions to z/OS 48
  Activating authorized functions (Beta SVC) in z/OS 49
Defining the subsystem ID to z/OS and initializing the subsystem 51
  IEFSSNxx entry for defining/initializing subsystem at IPL 52
  SETSSI command for defining/initializing subsystem dynamically 53
  BST01ARI: Initializing subsystem and security environment 54
  Security definitions required for subsystem initialization 57

Hardware and software requirements

Hardware

IBM z/Series (z/Architecture)

Operating system

Beta products run on the following IBM operating systems:

- Currently supported z/OS version

Beta products run under JES2 and JES3. No z/OS or JES system modifications are necessary.

Components

To install and run Beta products requires the following components, which must be compatible with your release of z/OS:

- SMP/E
- TSO/E
- ISPF and ISPF/PDF
- RACF (or an equivalent security system, for example, ACF2)
- VTAM (for multi-CPU operation only)

Pre-installation checklist

Overview

Before you begin the installation of Beta Systems Architecture, read through the following pre-installation checklist. The checklist tells you what information you will need during the installation process. The additional information required to install one of the Beta products can be found in the *Installation and System Guide* for that particular product.

Naming conventions

BSA and product dataset names used BSA and product documentation are used as examples only. You can choose names as you wish throughout the installation procedure.

Upgrading BSA V6 and Beta Systems V6 products

BSA V7 and Beta Systems V7 products require:

- A new CSI
- A new profile
- The BSA V7 SVC

You cannot use an existing V6 CSI, V6 profile, or V6 SVC for BSA V7-based products.

BSA components

The installation REXX of a product includes only the BSA components that are used by this product. Beta 23 is only included if it is used by the product. Use the setup program of the product that includes all shared components that you need in your environment.

	Beta23 (Browser)	Beta07 (BOF)	Beta09 (VDF)	Beta11 (CAF)	Beta12 (IAF)	Beta14 (VPF)
_beta access V7	–	–	X	X	X	–
_beta access monitor V7	X	–	–	–	–	–
_beta check|z V7	X	–	–	–	–	–
_beta log|z V7	X	–	X	X	X	–
_beta doc|z V7	X	X	X	X	X	X
_beta doc|z plus V7	X	–	X	X	X	–
_beta doc|z transform V7	X	–	X	X	X	–
_beta move V7	X	–	–	–	–	–

Note: _beta browse (Beta23) is not a BSA component, but it is installed in the BSA libraries because its functions are used by more than one Beta product.

Checklist

You can make the installation process shorter and easier by obtaining the following information beforehand.

1. For transfer of installation files to z/OS:
   - High-level qualifiers to be used by the extract job when setting up the libraries that will be used later in the installation process
2. If you do not use System Managed Storage (SMS), the following information is needed:
   - The type of **DASD units** you want to use for installation libraries.
   - The **volume names** on which the distribution libraries, target libraries, and SMP/E datasets are to be allocated. Different volumes can be used for the distribution libraries, target libraries and SMP/E datasets.

     - Target libraries:

     - Distribution libraries:

     - SMP/E datasets:
   - The **volume name** on which the Beta parameter library is to be allocated.
3. The **dataset name first level qualifier** for SMP/E and SMP/E service-dependent datasets.
4. The **dataset name first level qualifier** for the BSA distribution and target libraries.
5. The **dataset name** for the Beta parameter library (BETA.PARMLIB). This dataset cannot be the library SYS1.PARMLIB.
6. A **user type 3 or 4 SVC number** between 200 and 255 for the Beta SVC.

   For the Beta SVC to be loaded at IPL, the SVC has to be linked into the SYS1.LPALIB and specified in the active **IEASVC*xx*** member of the SYS1.PARMLIB.

   You must designate one SVC number per BSA environment (version) you will be installing.
7. The dataset name of a **system procedure library** into which the Beta SMP/E installation procedure will be placed.
8. The space requirements as specified in the *Release Notes and Update Instructions*.
9. The Beta installation procedure lets you install two different load libraries: the default names are BETA.APFLOAD for all modules that need APF authorization, and BSA.LOAD for all the remaining BSA modules. Every product installation procedure installs its own library (BETA*nn*.LOAD) in addition.

Modules requiring APF authorization

Modules requiring APF authorization

The following BSA modules are placed into the BETA.APFLOAD:

BST01ARI	Security initialization
BST00STH	Security router
BST00ATH	APF environment checker
BST01SSI	Subsystem initialization
BST01CMD	Operator commands for subsystem initialization
BST01SFF/ BST01RFF	Subsystem Function Facility
BST01SVC	Beta SVC module
BST01MST	Dynamic SVC update module
BST01XCF	BSA XCF Global Connect
BST00XIN	Default logon exit
BST09XIN	VDF user security initialization/termination (only if VDF is installed). Supports mixed-case passwords. Please see the Installation and System Guides for VAF/CAF/IAF.
B02UXSIN	BSA TCP/IP server logon exit (initialization/termination). Supports mixed-case passwords.

Additional modules are copied to this library during product installation, for example:

B*nn*UXSEC	product security exit
B*nn*SS*nnn*	SUBSYS interface module

Note on LOAD libraries

We suggest you follow the recommendation in the installation procedure and allocate three different load libraries when installing a Beta Systems product. The authorized BETA.APFLOAD should be put in the linklist. The program BST01ARI can then run at IPL time to initialize the security environment. The remaining two libraries (i.e. BSA.LOAD and BETA*nn*.LOAD) are concatenated under the STEPLIB DD statement of all product JCL samples.

Be aware of the following if you don't follow recommendations:

- If the same library is used for authorized and unauthorized modules (i.e. BETA.APFLOAD and BSA.LOAD are merged into one), BSA.LOAD must be APF-authorized. Because the BSA.LOAD is concatenated with the product BETA*nn*.LOAD in every product JCL, the BETA*nn*.LOAD needs to be authorized as well.
- If the BSA.LOAD library is not in the linklist concatenation, the program BST01ARI has to run as a batch job after every IPL for each Beta product subsystem. In this case, BST01ARI must not be specified in member IEFSSNxx as described in the Beta product Installation and System Guides. Only the subsystem ID must be entered here.

Overview of installation steps

Overview

This section gives an overview of the installation steps. Each step is then described in detail in a separate section of its own.

The installation descriptions in the BSA and product *Installation and System Guide* assume that you use one common SMP/E environment for the BSA functions and all Beta products. They also assume that you use only one set of BSA libraries for all Beta products.

Note: Always check the accompanying *Release Notes and Update Instructions* or Technical Note for possible instruction updates **before** beginning the installation process.

Installation steps

Step 1: Preparing the installation datasets

Download and unzip the installation archive on a Windows PC and then run the setup program. The product and/or BSA installation datasets are transferred to the z/OS host. The installation datasets are unpacked by extract jobs.

Step 2: Running the installation REXX

The installation REXX uses ISPF panels to ask you to enter the installation parameters. If you decide to install one or more Beta products in this installation process (provided they are on the same installation medium), the product installation procedure is called in the course of execution.

The installation procedure creates several batch jobs that are submitted in the next step. These batch jobs are placed in the newly allocated CNTL file.

Step 3: Submitting the installation batch jobs

You have to submit the installation batch jobs manually. Check that all batch jobs run correctly. If they don't, consult the output listings and try to determine the source of the error. If you cannot determine and correct the error, contact Beta Systems support. This step will place the Beta SVC into an LPALIB dataset.

Step 4: Specifying authorized functions to z/OS

The Beta SVC and the APF-authorized library are specified to your z/OS system. The activation of the Beta SVC and the APF-authorized library can be done either dynamically or via system IPL.

Note

- Perform one installation step after the other to make sure that the values you need are available. Step 1 must be executed before step 2, and so on.
- If you are installing a Beta product in a multi-CPU environment and would like to allow users access to Beta product subsystems on remote CPUs, you must set up the communication facilities and product subsystem definitions. You can use XCF (see "Multi-CPU using BSA XCF in a sysplex" on page 116) and/or OCF (see "Multi-CPU with the Open Communication Facility (OCF)" on page 86).

Step 1: Preparing the installation datasets

Standard installation medium

Beta Systems distributes its software via download from the Beta Systems website. BSA is included in the product download.

You will need to register when visiting this page for the first time:

`https://customer.betasystems.com/`

The software is made available as a ZIP archive, which includes all the folders and files that are necessary to install the software. Unzip the installation archive under Windows, and then run the setup program of the product, which transfers the installation datasets to the z/OS host. Folder names and file names are self-explanatory.

Important: The installation REXX of a product includes only the BSA components that are used by this product. Use the setup program of the product that includes all shared components that you need in your environment. See "Pre-installation checklist" on page 30 for more details.

Step overview

The installation process involves these steps:

1. Downloading and unzip the installation archive on a Windows PC.
2. Running the setup program on the Windows PC with automatic transfer of Beta Systems Architecture and/or BSA installation datasets to the z/OS host.
3. Unpacking the installation datasets on the z/OS host.
4. Running the Beta installation REXX on the z/OS host with tailoring of BETAnn.CNTL and BSA.CNTL.
5. Submitting the installation jobs for BSA and/or Beta Systems Architecture according to your installed level.

Each step is described in more detail below.

Requirements

By default, the setup program uses ftp for data transfer to the host, which assumes that IBM FTP server is available.

About 250 MB of temporary space are required on the PC. Space requirements on the z/OS host depend on the number of components you will be installing.

Installation steps on Windows PC

The following steps are carried out on a Windows PC:

1. Download and unzip the installation archive on a Windows PC.
2. Locate the setup program in the unpacked installation archive.

   You can find the setup program in the product **zOS** folder. All data for both the product and BSA are included in this setup program.
3. Run the setup program.
4. Provide the information requested by the setup program.

   The setup program will request the following information in sequence:

   - Which CD installation datasets are to be transferred to the host: CD installation datasets for BSA or the product or both?
   - Job card for the CD installation jobs CDUNPK*nn* and CDUDEL*nn*

     CDUNPK00 and CDUNPKnn unpack the installation datasets. CDUDEL00 and CDUDELnn can be used for deleting installation libraries and datasets. The job card for the BSA and product installation jobs will be requested later when you run the installation REXX on the z/OS host.
   - High-level qualifier for BSA and product installation datasets

     The high-level qualifier of BSA must be different from the high-level qualifier of the product. The high-level qualifiers specified here are used for transferred installation datasets and unpacked datasets. Specify new unique HLQs to avoid allocation errors during unpacking. The high-level qualifiers for SMP/E and tailored CNTL libraries, databases, etc. will be requested later when you run the installation REXX on the z/OS host.
   - Volser and device type for the installation datasets or SMS

   For automatic transfer via ftp:

   - Numeric or symbolic IP address of the z/OS host and ftp port number (default: 21)
   - Credentials (The password must be entered twice.)

   Alternatively, you can also transfer the datasets manually (see page 38).

Data transfer

The setup program transfers the installation datasets to the z/OS host.

Depending on whether you selected BSA or product or both, the following datasets will be present after transfer:

Dataset	Description
hlqbsa.CDFILE	Packed dataset (super file) that contains the BSA installation data
hlqbsa.JOBLIB	Partitioned dataset that contains the installation jobs CDUNPK00 and CDUDEL00
hlqprod.CDFILE	Packed dataset (super file) that contains the product installation data
hlqprod.JOBLIB	Partitioned dataset that contains the installation jobs CDUNPK*nn* and CDUDEL*nn*

Manual file transfer to the z/OS host

If you are experiencing difficulties with the setup program's automatic file transfer, you can also transfer the required files manually. To do this, choose option **Transfer files to z/OS manually** in the setup program and specify a destination folder for the files that are to be transferred.

The setup program creates the ftp script **ftp.txt**, which contains all the necessary information on the files that need to be transferred (source files, target datasets and their attributes, and transfer modes). You can run this script directly in your ftp client if you provide appropriate values for the placeholders <IPADDRESS>, <PORT>, <USERNAME> and <PASSWORD>.

For example, under Windows, open a command box, change to the destination folder, and then enter the following command:

```
ftp -s:ftp.txt
```

You can also use another suitable file transfer facility instead of ftp.

Important: For security reasons, you should delete the contents of the destination folder from your PC after the transfer, or at least remove your credentials from **ftp.txt**.

Unpacking the super file(s)

Unpack the super file(s) on the z/OS host:

- *hlqbsa*.JOBLIB(CDUNPK00) extracts installation data from the BSA super file *hlqbsa*.CDFILE.
- *hlqprod*.JOBLIB(CDUNPK*nn*) extracts installation data from the product super file *hlqprod*.CDFILE.

The result will be a collection of installation libraries and datasets whose format is usable as input for the installation process. For example, *hlqbsa*.DATA and *hlqprod*.DATA are libraries whose members contain the MCS statements (one per product and per additional facility or add-on).

If extraction leads to allocation errors

Job CDUNPK*nn* will end in error if it encounters a dataset of the same name during data extraction, for example, relicts of a previous or incomplete installation. In this case, run CDUDEL*nn* first to delete offending datasets.

- *hlqbsa*.JOBLIB(CDUDEL00) deletes all extracted BSA installation datasets.
- *hlqprod*.JOBLIB(CDUDELnn) deletes all extracted Beta Systems Architecture installation datasets.

You can ignore RC=8 if this is caused by message "IDC3012I ENTRY *datasetname* NOT FOUND", which means that one or more of the datasets to be deleted were not present. Afterwards, run CDUNPK*nn* to unpack again.

Step 2: Running the installation REXX

Overview

In step 2 you run the installation REXX to create the tailored JCL dataset that contains installation jobs and batch utilities.

You must run this REXX if you are a new customer, or if you are installing a new level of BSA or a Beta product, or if you are installing an additional system, for example, a test system.

Note

Always refer to the accompanying *Release Notes and Update Instructions* or *Technical Note* when you install a new level or apply maintenance to BSA or a Beta product.

Installation REXX

Under ISPF option 6, enter the appropriate TSO EXEC command to start the installation REXX.

Choose the installation REXX from the appropriate library depending on which installation jobs are to be tailored.

- If only BSA:

  ```
  EXEC 'hlqbsa.REXX(INSTALL)'
  ```

- If Beta product with BSA:

  ```
  EXEC 'hlqprod.REXX(INSTALL)'
  ```

Follow the instructions displayed by the installation REXX. After you have provided all the requested information, the installation REXX will tailor a new BETA*nn*.CNTL and/or BSA.CNTL from which you will submit the installation jobs required for your level.

Note: Don't interrupt the REXX after the tailoring of the product jobs. Follow instructions until the end to ensure that BSA jobs are tailored as well.

Getting help

The procedure named INSTALL will ask you for the information needed to tailor BSA to your installation. Press the HELP PF key if you would like an explanation of what you are being asked to do. You can use an ISPF split screen to execute an ISPF/PDF function.

Going back to check

If you wish to return to the first panel in the installation procedure, either to see what you entered or to change the data you entered, then type the word **BEGIN** in the command line of the current panel. If you wish to return to a previous panel in the installation procedure, either to see what you entered or to change the data you entered, then type the word **BACK** in the command line of the current panel.

Discontinuing installation

If you want to interrupt and discontinue the installation process, press the END PF key or type the word **END** in the command line. In this case, however, you must restart from the beginning of installation step 2.

Separate BSA environments

BSA V7 and BSA V7-based products require:

- A new SMP/E environment (CSI, SMP/E files, BSA target and distribution libraries)
- A new profile
- The BSA V7 SVC

We recommend that you use the same BSA V7 environment for all Beta Systems products that run under BSA V7.

BSA V7 is compatible with both BSA V7-based products and BSA V6-based products.

Installing Beta products

If you decide to install one or more Beta products during execution of the BSA installation procedure, the product installation procedure is called by the BSA procedure and the corresponding product batch jobs necessary for installation are generated. The product procedure then returns control to the BSA procedure.

Beta products available for selection

The only products you can select for installation are those that are on the same installation medium as the installation REXX. An installation REXX from a different installation medium must be executed completely one step after the other (starting with step 1). This is because the installation REXX contains a description of the contents of the installation medium. If you do not run the whole procedure, errors may occur.

Tailored batch jobs

The last part of the installation REXX tailors JCL for a series of batch jobs. These batch jobs are placed as members in the CNTL library.

You can use the same dataset for the tailored members of the Beta product and BSA (recommended). Alternatively, you can also use separate CNTL libraries.

Notes

- If you use only one CNTL for all products and BSA, the tailored BSA will be overwritten by the last installation process.
- Once a CNTL dataset is full, the installation procedure is suspended and a message appears. Use another terminal or ISPF split screen to compress the CNTL dataset. Afterwards, you can proceed with the installation procedure.
- The procedure also sets up a profile library where all the information entered during procedure execution is stored. This can be recalled when the procedure is executed again at a later time. Both the CNTL dataset and the profile library are created without an expiration date. Normally one common PROFILE library is sufficient.
- If you intend to install a separate BSA environment for a new Beta product, you must specify a separate profile library as well.

Step 3: Submitting installation batch jobs

Overview

In step 3, you submit the batch jobs that were created by the installation REXX during step 2.

Run each job separately, one after the other, checking the completion of each job before you run the next job. The installation of all BSA components cannot be assured if you do not run the jobs serially and with the proper completion codes.

Important: Check for possible instruction updates

Jobs I#BSAJ08 and I#BSAJ10 receive and apply new BSA PTFs. Always refer to the accompanying *Release Notes and Update Instructions* or *Technical Note* first for information on existing BSA PTFs and check for possible instruction updates.

BSA installation jobs

The BSA installation jobs I#BSAJ*xx* set up the SMP/E environment and install the BSA functions.

The BSA installation jobs only have to be run if you want to install a new BSA V7 environment. You have to run these jobs before you install the first BSA V7-based Beta product. If you have already installed a BSA V7-based Beta product, you should use the existing SMP/E environment and BSA libraries.

Following is a summary description of the BSA installation batch jobs. Each job is described in more detail later.

Member name	Purpose
I#BSAJ01	Allocates new SMP/E datasets
I#BSAJ02	Allocates the BSA target and distribution libraries on disk
I#BSAJ03	Allocates a SMP/E CSI for your new SMP/E environment
I#BSAJ04	Initializes a SMP/E CSI for your new SMP/E environment
I#BSAJ05	Copies the tailored SMP/E procedure to the specified system procedure library
I#BSAJ06	Sets up the zones of the SMP/E CSI
I#BSAJ07	Makes DD definitions in the global, target and distribution library zones for the datasets needed by SMP/E for the installation
I#BSAJ08	Receives the BSA functions and PTFs
I#BSAJ09	Applies and accepts the BSA functions
I#BSAJ10	Applies all accumulated BSA PTFs
I#BSAJ11	(Re)links the Beta SVC module into your environment (system LPALIB/APFLOAD library

Member name	Purpose
I#BSAJ12	Allocates the SFF parameter library (BETA.PARMLIB), which is used for all Beta product parameter members
I#BSAJ13	Copies the parmlib member B01LST*xx* for the Subsystem Function Facility (SFF) into the SFF parameter library (BETA.PARMLIB).

I#BSAJ01

Job I#BSAJ01 allocates new SMP/E datasets. Expect RC=0.

The following datasets are allocated:

DD name	Dataset name	Purpose
SMPLOG	BETA.SMPE.SMPLOG	SMP/E LOG dataset
SMPLOGA	BETA.SMPE.SMPLOGA	Alternate SMP/E LOG dataset
SMPMTS	BETA.SMPE.SMPMTS	SMP/E macro temporary store
SMPPTS	BETA.SMPE.SMPPTS	SMP/E program temporary store
SMPSTS	BETA.SMPE.SMPSTS	SMP/E source temporary store
SMPSCDS	BETA.SMPE.SMPSCDS	SMP/E control dataset

The description in this manual assumes that you use one SMP/E environment for all BSA 7-based products, separate from the z/OS SMP/E environment.

I#BSAJ02

Job I#BSAJ02 allocates the BSA distribution and target libraries. Expect RC=0.

The following datasets are allocated:

DD name	Dataset name	Purpose
BSALOAD	BSA.LOAD	Target BSA load module library
ABSALOAD	BSA.ALOAD	Distribution BSA load module library
BSAMLIB	BSA.ISPMLIB	Target BSA ISPF message library
ABSAMLIB	BSA.AISPMLIB	Distribution BSA ISPF message library
ISPTLIB	BSA.ISPTLIB	Target ISPF Table Library
AISPTLIB	BSA.AISPTLIB	Distribution ISPF Table Library
BSASAMP	BSA.SAMPLIB	Target BSA sample library
ABSASAMP	BSA.ASAMPLIB	Distribution BSA sample library
BSAPLIB	BSA.ISPPLIB	Target BSA ISPF panel library
ABSAPLIB	BSA.AISPPLIB	Distribution BSA ISPF panel library
BSASLIB	BSA.ISPSLIB	Target BSA skeleton library
ABSASLIB	BSA.AISPSLIB	Distribution BSA skeleton library
APFLOAD	BETA.APFLOAD	Library for APF authorized programs

The DD names listed above are only used in this installation job. They are not identical to the SMP/E DD definitions of job I#BSAJ07.

I#BSAJ03

Job I#BSAJ03 allocates an SMP/E CSI for your new SMP/E environment. Expect RC=0.

I#BSAJ04

Job I#BSAJ04 initializes the SMP/E CSI. Expect RC=0.

I#BSAJ05

Job I#BSAJ05 copies the tailored SMP/E procedure to your system procedure library. This procedure is called by most of the following installation jobs, which have been tailored accordingly. Job I#BSAJ05 specifies the member name and the procedure library name you provided when running the installation REXX. If a member of the same name already exists in the procedure library, this member will be replaced. Expect RC=0.

Before you run job I#BSAJ05:

Unless you want to replace an existing member, make sure that your procedure library does not already contain a member of the same name.

I#BSAJ06

Job I#BSAJ06 sets up the zones of the SMP/E CSI. Expect RC=0.

I#BSAJ07

Job I#BSAJ07 creates DD definitions in the global, target and distribution zones for the datasets needed by SMP/E for installation. Expect RC=0.

Following is a list of the DDDEFs created for the BSA and system libraries.

Target zone:

Dataset name	**DDDEF**
BSA.LOAD	BSTLOAD
BSA.ISPPLIB	BSTPLIB
BSA.ISPMLIB	BSTMLIB
BSA.ISPSLIB	BSTSLIB
BSA.ISPTLIB	BSTTLIB
BSA.SAMPLIB	BSTSAMP
BETA.APFLOAD	BSTAPF
CEE.SCEELKED	SCEELKED
CEE.SCEELKEX	SCEELKEX
CEE.SCEECPP	SCEECPP

Dlib zone:

Dataset name	**DDDEF**
BSA.ALOAD	ABSTLOAD
BSA.AISPPLIB	ABSTPLIB
BSA.AISPMLIB	ABSTMLIB
BSA.AISPSLIB	ABSTSLIB
BSA.AISPTLIB	ABSTTLIB
BSA.ASAMPLIB	ABSTSAMP

I#BSAJ08

Job I#BSAJ08 receives the functions and the accumulated PTFs of BSA. Expect RC=0.

Table of SYSMOD entries

	Function	PTFs	APARs	Description
BSF	RBS7100	PBS....	TBS....	Base System Facility
SFF	RBS7100	PBS....	TBS....	Subsystem Function Facility
OCF	RBS7100	PBS....	TBS....	Open Communication Facility
DMF	RBS7100	PBS....	TBS....	Data Management Facility
BOF	RBS7100	PBS....	TBS....	Base Output Facility
BAF	RBS7100	PBS....	TBS....	Base Archive Facility
RPG	RBS7100	PBS....	TBS....	_beta report
VDF	RBS7100	PBS....	TBS....	VTAM Dialog Facility
CAF	RBS7100	PBS....	TBS....	_beta caf
IAF	RBS7100	PBS....	TBS....	_beta iaf
VAF	RBS7100	PBS....	TBS....	_beta vaf
BSM	RBS7100	PBS....	TBS....	BSA Service Manager

I#BSAJ09

Job I#BSAJ09 applies and accepts the BSA functions.

The CAF and IAF modules must be linked into the appropriate load library (CICS or IMS) after the installation of _beta caf and _beta iaf. B11LINK and B12LINK are sample jobs in the BSA.SAMPLIB.

I#BSAJ10

Job I#BSAJ10 applies all BSA accumulated PTFs.

Following is a list of the standard SMP/E return codes for this job:

0 PTFs applied.

4 PTFs applied. This return code is typically okay. You can find more information in the accompanying Release Notes and Update Instructions or Technical Note.

8 Error. First check whether there are any PTFs with the SMP/E parameter `++HOLD... SYSTEM REASON(...)`. You can find this information with additional instructions in the accompanying Release Notes and Update Instructions or Technical Note. Otherwise, please contact Beta Systems support.

12 No PTFs for FMID. Check the accompanying release notes to find out whether there are any PTFs. RC=12 is okay if there are no PTFs.

I#BSAJ11

This LKED job links or relinks the Beta SVC module (BST01SVC) into your environment (system LPALIB / APFLOAD dataset) under the name BST01SVC. Make sure that the library (DD name SYSLMOD) matches your installation naming conventions.

The Beta SVC is used for communication between TSO users, batch jobs, and the corresponding Beta subsystem. The SVC enables modifications to be made to the database from different address spaces without any integrity problems.

The Beta SVC must be specified and activated in z/OS. For more information, see:

- "Step 4: Specifying authorized functions to z/OS" on page 48
- "Activating authorized functions (Beta SVC) in z/OS" on page 49

I#BSAJ12

Job I#BSAJ12 allocates the SFF parameter library (BETA.PARMLIB), which is used for the parameter members for all Beta products. The BETA.PARMLIB must be a PO dataset with a logical record length of 80.

Dataset name	Purpose
BETA.PARMLIB	Beta parameter library

I#BSAJ13

Job I#BSAJ13 copies the parmlib member B01LST*xx* into the Beta parameter library (BETA.PARMLIB).

Member B01LST*xx* is tailored for the Subsystem Function Facility (SFF) according to your specifications during the installation REXX.

For more information, see "LST parameters in B01LSTxx" on page 69.

Step 4: Specifying authorized functions to z/OS

Overview

After job I#BSAJ11 has (re)linked the Beta SVC into the SYS1.LPALIB or BETA.APFLOAD, the Beta SVC must be specified and activated in z/OS.

Procedure

1. Specify the Beta SVC to z/OS.

   Enter the Beta SVC in member IEASVC*xx* of the SYS1.PARMLIB in the format:

   ```
   SVCPARM nnn,REPLACE,TYPE(4),APF(NO),EPNAME(BST01SVC)
   ```

   (where *nnn* must be replaced with the SVC number you designated. For TYPE enter either 3 or 4.)

   Note: The dataset containing the Beta SVC must be available in the LPALIB system concatenation at IPL time. In member IEALPA*xx* of your SYS1.PARMLIB, enter an appropriate statement that includes the library name and the module name of the Beta SVC.

2. Specify the BETA.APFLOAD to z/OS.

   Add the name your BETA.APFLOAD to the list of APF-authorized libraries in member IEAAPF*xx* or PROG*xx* of your SYS1.PARMLIB.

   The BETA.APFLOAD is normally allocated as a separate library during BSA installation. All modules that require APF authorization are copied into this load library during installation.

   Note: If you specify your BETA.APFLOAD in member PROG*xx* of your SYS1.PARMLIB, make sure that it has been activated before starting any Beta product.

3. **Strongly recommended**: Include your BETA.APFLOAD in the linklist, which is defined in member LNKLSTxx of your SYS1.PARMLIB.

If BETA.APFLOAD is not in the linklist

We strongly recommend that you include your BETA.APFLOAD in the linklist.

Be aware of the following consequences if you decide **not** to include the BETA.APFLOAD in the linklist:

- BST01ARI cannot run at IPL time.
- Your BETA.APFLOAD must be concatenated in the STEPLIB of every Beta product batch job and started task JCL.
- All the other libraries in the STEPLIB concatenation of each Beta product batch job and started task JCL must also be APF authorized.

Activating authorized functions (Beta SVC) in z/OS

Overview

Two methods can be used to activate authorized functions after they have been specified to z/OS. Either activate the SVC in LPA by performing an IPL with CLPA (IPL method), or follow the instructions listed below to activate the SVC dynamically by running job SVCUPDTE (dynamic method).

IPL method

Perform an IPL with CLPA in order to activate the Beta SVC in LPA and/or the authorized library

Note: If you install one or more Beta products at the same time as BSA, remember to go through the product installation steps before performing the IPL, as an IPL also activates the product subsystem entries.

Dynamic method

1. Stop all Beta product subsystems, started tasks and batch jobs that are using this SVC number.
2. If z/OS has not yet recognized the BETA.APFLOAD as an APF-authorized library, define this load library to z/OS as follows:
   1. Add an entry for the BETA.APFLOAD to the PROG*xx* member of the SYS1.PARMLIB.
   2. Issue the operator command `SET PROG=xx` to activate this change.
3. Modify the SVCUPDTE job to match your installation naming conventions. Make sure that the SVC number in the PARM field contains the correct number of the Beta SVC you designated.

   The sample job is found in member SVCUPDTE in the BSA.CNTL:

```
+--------------------------------------------------------------------+
|//INIT   EXEC PGM=BST01MST,                                          |
|//       PARM='REPLACE,212'                                          |
|//*      PARM='DELETE,212'                                           |
|//*                                                                  |
|//*STEPLIB DD  DISP=SHR,                                             |
|//*            DSN=BETA.APFLOAD                                      |
|//*                                                                  |
|//LOADLIB  DD DISP=SHR,                                              |
|//            DSN=BETA.APFLOAD                                         |
|//SYSUDUMP DD SYSOUT=*                                               |
+--------------------------------------------------------------------+
```

4. Submit the modified SVCUPDTE job to dynamically activate or update the Beta SVC.

EXEC parameter

REPLACE,*nnn*	Replaces the Beta SVC (*nnn* is the SVC number)
DELETE,*nnn*	Deletes the Beta SVC (*nnn* is the SVC number)

DD statements

LOADLIB	APF-authorized load library from which the Beta SVC is loaded

Return codes

0 Beta SVC successfully replaced or deleted

4 This return code can have the following causes:

- RACF resource not defined
- Deleted SVC has been used for the initialization of one or more existing subsystems, which are no longer useable; warning message 9109W is output for each affected subsystem

8 Syntax error

12 This return code can have the following causes:

- Not enough storage
- CSA is full

16 This return code can have the following causes:

- Cannot find/load the Beta SVC member
- Open error in Beta SVC library
- ESTAE failure
- Task not APF-authorized
- No input parameter specified

20 Task abended

RACROUTE macro

To prevent unauthorized access to the program SVCUPDTE, the RACROUTE macro is issued when a user submits this job. The RACF request is:

```
REQUEST=AUTH,CLASS='FACILITY',ACCESS=READ
```

The entity name is BETA.SVC.*keyword.nnn*, where *keyword* is either REPLACE or DELETE, depending on the SVCUPDTE function to be performed, and *nnn* is the Beta SVC you designated.

What to do next

You have completed steps 1 to 4 above, and BSA has been successfully installed. Your BSA environment has been set up with the default options and you can now customize it.

Defining the subsystem ID to z/OS and initializing the subsystem

Overview

Before you can start a Beta *nn* started task:

1. Its subsystem ID must be defined to z/OS.
2. Its subsystem environment and security environment must be initialized by BST01ARI.

You can include the BST01ARI call in the entry or command that defines the subsystem. Or you can define the subsystem first and then run BST01ARI some time later (but before the start of the STC).

Use the profile `BETA.INIT.`*`ssid`* in the RACF facility class to protect BST01ARI against unauthorized access (see "RACF authorization for BST01ARI" on page 55).

Prerequisites for initialization

- The required security definitions must be present (see "Security definitions required for subsystem initialization" on page 57).
- You must be authorized to run the procedure.
- The subsystem you want to initialize must not be active.
- BST01ARI, BST01SSI and the modules specified in the B*nn*SSI*xx* member (see page 56) must be in an APF-authorized library link-edited with AC(1). The default name of this library is BETA.APFLOAD. This library must be included in the linklist if BST01ARI is called at IPL or via SETSSI.

IPL or dynamic methods

The subsystem can be defined to z/OS at IPL or dynamically. There are several options:

- At IPL via an entry in the IEFSSNxx member

  If you include the BST01ARI call in your IEFSSN*xx* entry, the subsystem is defined and initialized at IPL. **Note**: BST01ARI must be available in the linklist.

  Alternatively, you can create an IEFSSNxx entry without this call and run BST01ARI later.

- Dynamically via SETSSI

  If you include the BST01ARI call in your SETSSI command, the subsystem is defined and initialized when you run the command. **Note**: BST01ARI must be available in the linklist.

  Alternatively, you can submit the SETSSI command without this call and run BST01ARI later.

- Dynamically via BST01ARI

  If BST01ARI detects that a subsystem has not been defined, it defines this subsystem ID to z/OS and then initializes the subsystem environment and security environment. **Note**: This is protected by higher security (see "RACF authorization for BST01ARI" on page 55).

IEFSSNxx entry for defining/initializing subsystem at IPL

Overview

Create an appropriate entry in the IEFSSN*xx* member of your SYS1.PARMLIB if the subsystem is to be defined to z/OS at IPL.

If you include the BST01ARI call in your IEFSSN*xx* entry, the subsystem is defined and initialized at IPL.

IEFSSNxx entry

Use this syntax when creating an entry with BST01ARI call in the IEFSSN*xx* member:

```
SUBSYS   SUBNAME(ssid)
         INITRTN(BST01ARI)
         INITPARM('parmlib(member)')
```

where *ssid* is the Beta *nn* subsystem ID. The INITRTN parameter specifies the program to be executed (BST01ARI), and the INITPARM parameter specifies the B*nn*SSI*xx* member in the BETA.PARMLIB, which contains the parameters to be used by BST01ARI (see "Parameters for BST01ARI in BnnSSIxx" on page 56).

Result: The specified subsystem is defined and initialized at IPL.

Use this syntax if you only want the subsystem to be defined to z/OS, but without initialization:

```
SUBSYS   SUBNAME(ssid)
```

Result: The specified subsystem is defined at IPL. You have to run BST01ARI at some later time to initialize the subsystem.

Example

The following example shows an IEFSSNxx entry for a _beta log|z system. The subsystem is defined and initalized at IPL.

```
SUBSYS   SUBNAME(B92P)
         INITRTN(BST01ARI)
         INITPARM('BETA.PARMLIB(B92SSI00)')
```

Address space

A separate address space is started for initialization by system procedure IEESYSAS when the subsystem is initialized at IPL.

The start command is as follows:

```
IEESYSAS.BETAssid,PROG=BST01ARI,PARM='ssid,parmlib(member))'
```

The address space has the name **BETA*ssid***. Among other things, this procedure ensures that the specified parmlib member can be in a library that can be held in the user catalog and it can be on a migrate volume.

SETSSI command for defining/initializing subsystem dynamically

Overview

The operator command SETSSI enables you to define a subsystem ID dynamically to z/OS.

If you include the BST01ARI call in your SETSSI command, the subsystem is defined and initialized.

SETSSI command

Use this syntax if you want to include the BST01ARI call in your SETSSI command:

```
SETSSI ADD,SUB=ssid,INITRTN=BST01ARI,INITPARM='parmLib(member)'
```

where *ssid* is the Beta *nn* subsystem ID. The INITRTN parameter specifies the program to be executed (BST01ARI), and the INITPARM parameter specifies the B*nn*SSI*xx* member in the BETA.PARMLIB, which contains the parameters to be used by BST01ARI (see "Parameters for BST01ARI in BnnSSIxx" on page 56).

Result: The specified subsystem is defined and initialized when you run the command.

Use this syntax if you only want the subsystem to be defined to z/OS, but without initialization:

```
SETSSI ADD,SUB=ssid
```

Result: The specified subsystem is defined when you run the command. You have to run BST01ARI at some later time to initialize the subsystem.

Example

The following example shows a SETSSI command for a _beta log|z system. The subsystem is defined and initalized when you run the command.

```
SETSSI ADD,SUB=B92T,INITRTN=BST01ARI,INITPARM='BETA.PARMLIB(B92SSI01)'
```

Separate address space for initialization

A separate address space is started by system procedure IEESYSAS when the subsystem is initialized via the SETSSI command.

The start command is as follows:

```
IEESYSAS.BETAssid,PROG=BST01ARI,PARM='ssid,parmLib(member))'
```

The address space has the name **BETA*ssid***. Among other things, this procedure ensures that the specified parmlib member can be in a library that can be held in the user catalog and it can be on a migrate volume.

Note

It is not possible to use SETSSI to activate or deactivate an existing subsystem that was already defined to z/OS beforehand.

BST01ARI: Initializing subsystem and security environment

Overview

BST01ARI initializes the subsystem environment and security environment of a subsystem. BST01ARI must have run before you can start the STC.

The BST01ARI call can be included in the IEFSSN*xx* entry or SETSSI command that defines the subsystem to z/OS. Or you can call BST01ARI in a batch job or the STC procedure.

BST01ARI in batch job or STC

Use this syntax when calling BST01ARI in a batch job or the STC procedure:

```
//stepname EXEC PGM=BST01ARI,
//         PARM='ssid,parmlib(member)'
```

where *ssid* is the Beta *nn* subsystem ID. The EXEC parameter specifies the B*nn*SSI*xx* member in the BETA.PARMLIB, which contains the parameters to be used by BST01ARI (see "Parameters for BST01ARI in BnnSSIxx" on page 56).

Example

```
//stepname EXEC PGM=BST01ARI,
//         PARM='B92T,BETA.PARMLIB(B92SSI01)'
```

If *ssid* not defined

If BST01ARI runs in a batch job or STC procedure and detects that the specified subsystem has not been defined, it defines this subsystem ID to z/OS and then initializes the subsystem environment and security environment.

Note: This is protected by higher security (see "RACF authorization for BST01ARI" on page 55).

Reinitialize environment

If you have to run BST01ARI to reinitialize the subsystem, for example, because the security exit has been changed, you must stop the started task first before running BST01ARI.

Address space

The complete initialization functionality is supported when BST01ARI runs in a batch job or the STC procedure.

No additional address space is created.

RACF authorization for BST01ARI

Use the following profile in the RACF facility class to protect BST01ARI against unauthorized access:

```
BETA.INIT.ssid
```

where *ssid* is the subsystem ID.

The following RACROUTE is issued when a user calls BST01ARI to initialize a subsystem that has been defined to z/OS:

```
REQUEST=AUTH,CLASS='FACILITY',ACCESS=READ
```

If this profile is not defined, the result depends on your security system. RACF is typically set up to allow the requested action anyway, but returns a warning. In this case, BST01ARI initializes the subsystem and ends with RC=4.

The following RACROUTE is issued when a user calls BST01ARI to define and initialize a subsystem:

```
REQUEST=AUTH,CLASS='FACILITY',ACCESS=ALTER
```

If this profile is not defined, BST01ARI ends with RC=16.

Return codes of BST01ARI

0 The program terminated normally

4 The program terminated with a warning. Possible reasons are:

- Profile BETA.INIT.*ssid* has not been defined
- A module which was specified could not be loaded, because it was not found or because it was not in an authorized library
- FREEMAIN failed

8 The program terminated abnormally due to parse error. Statements in member IEFSSNxx are not correctly coded.

12

- The program terminated abnormally because GETMAIN failed.
- The SVC you have specified is invalid for Beta products. The corresponding SVC entry for the subsystem is deleted.

16 The program terminated abnormally. Possible reasons are:

- ENQUEUE failed (for example, started task was active)
- DYNALLOC or OPEN error (for example, parameter dataset or member not found)
- ESTAE failed (recovery)
- BST01ARI and BST01SSI are not authorized.

20 The program abended.

Parameters for BST01ARI in BnnSSIxx

Member B*nn*SSI*xx* in the BETA.PARMLIB contains a set of parameters for the program BST01ARI. These parameters can be coded as shown in the table below. See the product documentation for details on which parameters are to be used and how to use them.

Parameter	Description	Option
UXSRT	Security router (module name must be BST00STH)	required
UXSEC	Security user exit module Specify UXSEC=IEFBR14 if the security exit is not required.	optional
UXSIN	Security user exit module for logging onto a subsystem If security is not required, simply do not code this parameter. Do not attempt to switch it off with IEFBR14, as this will return a warning. (Exception: BOF (see "Running BOF as a separate started task" on page 282))	optional
SVC	Beta SVC number	required
XCF	YES activates XCF functionality (see "Multi-CPU using BSA XCF in a sysplex" on page 116) See also *Installation and System Guide* of the product concerned.	optional
XCF_PROD	Two-digit product identifier of the subsystem This parameter is required if XCF = YES. See also *Installation and System Guide* of the product concerned.	optional
XCF_GBAS	Enables the use of the BSA XCF Global Connect component (see page 131) See also *Installation and System Guide* of the product concerned.	optional
GOTO_OCF	Enables communication between an XCF sysplex and an external system (see "Multi-CPU using BSA XCF in a sysplex" on page 116) See also *Installation and System Guide* of the product concerned.	optional
ROUTE_TO	Automatically reroutes requests to the specified subsystem See also *Installation and System Guide* of _beta access and _beta doc|z.	optional
SUBSYS	Specifies the name of the subsystem interface program The subsystem interface program ensures that jobs using the subsystem interface do not end abnormally when the subsystem is not available, but remain in wait status until the subsystem is started. See also *_beta doc|z Installation and System Guide*.	optional
FSSM	Specifies the functional subsystem module See also *_beta doc|z Installation and System Guide*.	optional

Security definitions required for subsystem initialization

Overview

Before you initialize a subsystem, the following additional definitions are required for your security system (normally RACF):

- Definition of a STARTED profile for the STC/user BETA*ssid* or equivalent in started procedures table (ICHRIN03)
- Definition of a user profile BETA*ssid* with the following access rights:
  - READ access for the STC/user BETA*ssid* to the parmlib that contains the initialization member to be used
  - READ access of the STC/user BETA*ssid* to the load libraries of the corresponding Beta product, including BSA

To grant the above-mentioned access rights to a user, we advise you to create a generic STARTED profile.

Example:

```
Profile  => BETA*.*
Class    => STARTED
UACC     => NONE

STDATA: USER= BETAINIT GROUP= BETAGRP TRUSTED= NO PRIVILEGED= NO TRACE= NO
```

This defines user BETAINIT / group BETAGRP with the corresponding READ access rights

Parameter SN=

The parameter **SN=** enables you to specify the name to be used as identifier when starting the system procedure IEESYSAS. The maximum length of the name is 8 characters. The first character must be alphabetic. If no value is coded, the default BETA*ssid* is used.

Coding the parameter **SN=** enables the following:

- To use a STARTED profile (for example, BETAINIT.*)
- To keep the number of required RACF definitions to a minimum when running numerous Beta product STCs (subsystems) in parallel

You can use parameter **SN=** when coding the required entries for Beta product subsystems in the IEFSSN*xx* member. When coded, this parameter overwrites the identifier BETA*ssid* in the start command with the specified value. SN= can also be used with operator command SETSSI.

SN= in IEFSSN*xx*

The extension for entries in the IEFSSN*xx* member looks like this:

```
SUBSYS SUBNAME(ssid)                         /* BETA Subsystem ID        */
         INITRTN(BST01ARI) INITPARM('parmLib(member)[,SN=stcname]')
```

Example:

```
SUBSYS SUBNAME(BnnT)                         /* BETAnn                   */
         INITRTN(BST01ARI) INITPARM('BETA.PARMLIB(BnnSSI00),SN=stcname')
```

This will generate the following command:

```
IEESYSAS.stcname,PROG=BST01ARI,PARM='BnnT,BETA.PARMLIB(BnnSSI00)'
```

SN= with SETSSI:

This extension can also be used when a subsystem is initialized by means of operator command SETSSI:

```
SETSSI ADD,SUB=ssid[,INITRTN=BST01ARI,INITPARM='BETA.PARMLIB(BnnSSIxx)[,SN=stcname]']
```

Note on RACF message IEE296I

If RACF does not find a matching STARTED profile, message IEE296I may be generated with corrupt values. IBM provides APAR (OA36604), which ensures that the values in the message are correct. We advise you to apply this APAR:

```
URL:  https://www-304.ibm.com/support/docview.wss?uid=isg1OA36604&crawler=1&wv=1
```

Example of IEFSSNxx member

This example defines several Beta product systems in the IEFSSNxx member using BEGINPARALLEL, and initializes the subsystems with the corresponding parameter values:

```
.....
BEGINPARALLEL
.....
SUBSYS SUBNAME(B07T)               /* BETA07                     */
       INITRTN(BST01ARI) INITPARM('BETA.PARMLIB(B07SSI00)')
SUBSYS SUBNAME(B88T)               /* BETA88                     */
       INITRTN(BST01ARI) INITPARM('BETA.PARMLIB(B88SSI00),SN=')
SUBSYS SUBNAME(B09T)               /* BETA09                     */
       INITRTN(BST01ARI) INITPARM('BETA.PARMLIB(B09SSI00),SN=B09STC')
SUBSYS SUBNAME(B92T)               /* BETA92                     */
       INITRTN(BST01ARI) INITPARM('BETA.PARMLIB(B92SSI00),SN=BETAINIT')
SUBSYS SUBNAME(B93T)               /* BETA93                     */
       INITRTN(BST01ARI) INITPARM('BETA.PARMLIB(B93SSI00),SN=BETAINIT')
SUBSYS SUBNAME(B97T)               /* BETA97                     */
       INITRTN(BST01ARI) INITPARM('BETA.PARMLIB(B97SSI00),SN=BETAINIT')
```

Customization

In this chapter

Topic	**Page**
Beta parameter library (BETA.PARMLIB)	60
Examples of started task (SFF) and batch job (RFF) procedures	61
Specifying parameters	63
Selection of parameter members	65
Parmlib member syntax	66
BSA LST parameters and how to change them	68
LST parameters in B01LSTxx	69
LST parameters in BnnLSTxx	71
Specifying different routing codes	74
Suppressing messages	76
Suppressing dumps for SFF/RFF and SFF storage regions	77
Static system symbol support	79

Beta parameter library (BETA.PARMLIB)

Overview

The installation defaults for Beta products are defined in members of the Beta parameter library.

The documentation uses the default name BETA.PARMLIB for this library. The members are called B*nn*LST*xx*.

BETA.PARMLIB

The BETA.PARMLIB is allocated during installation by the BSA installation job I#BSAJ12. The dataset name of this library is specified in the JCL of batch jobs and started task procedures in the SFFPARM DD name.

The dataset referenced by SFFPARM must be a PO dataset with a logical record length of 80. The relevant parameter members are selected from this library during program start according to the selection algorithm described in "Selection of parameter members" on page 65.

Note: Do **not** concatenate multiple datasets in the SFFPARM DD name. Only the first library is searched for the parameter member.

Note: Do **not** code a member name in the SFFPARM DD name. The name of the library must be coded for the selection algorithm to work.

LST parameters

The BETA.PARMLIB contains all installation parameter members for Beta products.

The members in the BETA.PARMLIB contain two different kinds of LST parameters:

- Parameters for the Subsystem Function Facility (SFF) and Remote Function Facility (RFF)

  SFF and RFF are BSA components used by all Beta products. The SFF is used for the product started task, and the RFF is used for the batch jobs which sign on to the started task. Only one global parameter member is necessary; it can be used in common by all the Beta products that are installed. The global SFF parameters are described in these sections:

  - "LST parameters in B01LSTxx" on page 69
  - "LST parameters in BnnLSTxx" on page 71
- Product-specific parameters

  The product-specific parameters are described in the Beta product *Installation and System Guide* in the "Customization" chapter.

Examples of started task (SFF) and batch job (RFF) procedures

Overview

This section provides sample procedures for Beta product started tasks (BST01SFF) and batch jobs (BST01RFF). These are tailored by the installation REXX and then copied to the specified procedure library using installation job I#B01J07.

Sample BST01SFF (started task)

```
+-----------------------------------------------------------------------+
|//*                                                                    |
|//BETA01   PROC  R=0M,TRACE=NO,LST=xx                                  |
|//*                                                                    |
|//BETA01   EXEC PGM=BST01SFF,                                          |
|//             PARM=('S=nn,B01LST=xx,BnnLST=&LST',                     |
|//             'BQL_TRACE=&TRACE'),                                    |
|//             TIME=1440,REGION=&R                                     |
|//*                                                                    |
|//STEPLIB  DD  DISP=SHR,                                               |
|//             DSN=BSA.LOAD                                            |
|//*        DD  DISP=SHR,                                               |
|//*            DSN=BETA.APFLOAD                                        |
|//         DD  DISP=SHR,                                               |
|//             DSN=BETAnn.LOAD                                         |
|//*                                                                    |
|//SFFPARM  DD  DISP=SHR,                                               |
|//             DSN=BETA.PARMLIB                                        |
|//*                                                                    |
|//BnnDEF   DD  DISP=SHR,                                               |
|//             DSN=BETAnn.DB.DEF                                       |
|...                                                                    |
+-----------------------------------------------------------------------+
```

Sample BST01RFF (batch job)

```
+-----------------------------------------------------------------------+
|//*                                                                    |
|//Bnnxxxxx PROC  R=0M,LST=xx                                           |
|//*                                                                    |
|//Bnnxxx   EXEC PGM=BST01RFF,                                          |
|//             PARM=('S=nn,B01LST=00,B01LST=&LST',                     |
|//             'PGM=Bnnxxxxx,SIGNON=YES'),                             |
|//             TIME=1440,REGION=&R                                     |
|//*                                                                    |
|//STEPLIB  DD  DISP=SHR,                                               |
|//             DSN=BSA.LOAD                                            |
|//*        DD  DISP=SHR,                                               |
|//*            DSN=BETA.APFLOAD                                        |
|//         DD  DISP=SHR,                                               |
|//             DSN=BETAnn.LOAD                                         |
|//*                                                                    |
|//SFFPARM  DD  DISP=SHR,                                               |
|//             DSN=BETA.PARMLIB                                        |
|//*                                                                    |
|//BnnDEF   DD  DISP=SHR,                                               |
|//             DSN=BETAnn.DB.DEF                                       |
|...                                                                    |
+-----------------------------------------------------------------------+
```

DD statements

DD name	**Description**
STEPLIB	BSA and Beta *nn* load libraries
SFFPARM	Beta parameter library
B*nn*DEF	The Beta *nn* database definition file or DUMMY for slave started tasks and batch jobs with SIGNON=YES. (Please see the product documentation for exceptions.)

Specifying parameters

Overview

Parameters can be specified at three different locations:

- In the EXEC statement
- In members B01LST*xx* of the BETA.PARMLIB
- In members B*nn*LST*xx* of the BETA.PARMLIB

You will find details below.

EXEC statement

In the EXEC statement of the started task procedure or any batch job, EXEC parameters can be passed to SFF, RFF or product functions. All keywords can be coded in the EXEC statement. You can also code 'B01LST=*xx'* to load a specific SFF parmlib member other than the default B01LST00, and/or 'B*nn*LST=*xx'* (substituting *nn* with the Beta product identifier) to load a parmlib member other than the default B*nn*LST00.

EXEC parameters

The following parameters must be specified in EXEC Parm:

Parameter	Description	Option	Default
S	Describes the product identifier. The 2-digit product number must be specified. This number must match the number specified in DD statement B*nn*DEF.	required	None
B01LST	Describes the SFF parmlib member B01LSTxx	optional	00
B*nn*LST	Describes the product parmlib member B*nn*LST*xx*	optional	00
SIGNON	Describes the type of communication for the started task and batch jobs. SIGNON=NO means that no other started task or batch job that processes the database can be active at the same time, i.e. this particular batch job or started task must function exclusively as the sole master. SIGNON=YES means that this started task or batch job can function as a slave and cannot carry out any database tasks of its own. In this case, an active master started task must always be available.	optional	NO

B01LST*xx* in BETA.PARMLIB

Parameters for SFF/RFF system startup are coded in member B01LST*xx* of the BETA.PARMLIB, where *xx* can be any two-digit/character combination (00..99, AA..ZZ).

For a list of parameters that can be coded in B01LST*xx*, see "LST parameters in B01LSTxx" on page 69.

B*nn*LST*xx* in BETA.PARMLIB

Product-specific parameters are coded in member B*nn*LST*xx* of the BETA.PARMLIB, where *nn* is the number of the product. *xx* can be any two-digit/character combination (00..99, AA..ZZ).

Order of precedence

When a parameter is coded more than once, the following order of precedence applies:

- Parameters in the EXEC statement take precedence over parameters coded in members B01LST*xx*/B*nn*LST*xx*.
- Parameters in the B01LST*xx* member take precedence over parameters in the B*nn*LST*xx* member.

  Note: Only selected parameters can be coded in both members.

Selection of parameter members

Overview

At SFF/RFF startup time, the program determines which parameter members are to be loaded.

During every startup, two members are loaded, **one** B01LST*xx* and **one** B*nn*LST*xx*. Parmlib members are selected as follows:

For B01LST*xx*:

If parameter **B01LST=*xx*** is coded in the EXEC statement, the specified member (B01LST*xx*) is loaded.

If parameter **B01LST=*xx*** is **not** coded in the EXEC statement, the default member B01LST00 is loaded.

For B*nn*LST*xx*:

If parameter **B*nn*LST=*xx*** is coded in the EXEC statement, the specified member (B*nn*LST*xx)* is loaded.

If parameter **B*nn*LST=*xx*** is **not** coded in the EXEC statement, the default member B*nn*LST00 is loaded.

Example

Default member	B01LST00, B*nn*LST00

In the following lines from the started task JCL, the EXEC parameter specifies both parameter members: therefore both the specified members, B01LST01 and B*nn*LST02, are loaded.

```
+--------------------------------------------------------------------+
|//BETASTC  EXEC PGM=BST01SFF,                                       |
|//              PARM='S=nn,B01LST=01,BnnLST=02'                     |
+--------------------------------------------------------------------+
```

If the EXEC parameter specifies only parameter B01LST=01 for SFF/RFF, members B01LST01 and B*nn*LST00 are loaded.

If the EXEC parameter in the started task JCL does not specify any parameter members, the default members B01LST00 and B*nn*LST00 are loaded.

Parmlib member syntax

Overview

This section describes the syntax for specifying LST parameters in members of the BETA.PARMLIB.

General syntax rules

LST parameters are name/value pairs. Use an equal sign (=) to separate the name (keyword) and the value.

```
keyword=value
```

Any number of blanks can be added to the left and to the right of the equal sign.

```
keyword    = value
```

You can code the name/value pair at any position between column 1 and 71 on the line. Column 72 is reserved for continuation.

Code each LST parameter on a separate line. The keyword and the value must be coded on the same line. Blank lines are ignored.

Comments

Code an asterisk (*) at column 1 to make the entire line a comment.

Comments can also be coded after the value on the same line. Separate the comment from the value by at least one blank. If a keyword does not require a value, no comment can be coded on this line.

In the following example, only the LST parameter of the first line is processed.

```
 keyword    = value                 optional comment
*keyword    = value                 optional comment
```

Values with blanks or other special characters

A value must be enclosed in quotation marks or parentheses if it includes blanks. All of the following are valid:

```
keyword    = 'value with blanks'   optional comment
keyword    = "value with blanks"   optional comment
keyword    = (value with blanks)   optional comment
```

Enclosing the value protects it against being partly interpreted as comment.

Also use quotation marks or parentheses for values containing other special characters. For example, if the value includes single quotation marks, enclose it in double quotation marks. If the value includes double quotation marks, include it in single quotation marks.

Subparameters

Some values consist of several comma-separated subparameters. These values are often enclosed in parentheses or quotation marks for better readability, even if the value does not include blanks. Add a comma for required positional subparameters. All of the following are valid:

```
keyword    = sub1,sub2,,sub4        optional comment
keyword    = 'sub1,sub2,,sub4'      optional comment
keyword    = "sub1,sub2,,sub4"      optional comment
keyword    = (sub1,sub2,,sub4)      optional comment
```

Continuation

Long values can be coded on multiple lines. Use the following syntax:

- Code any non-blank character at column 72 to indicate that the value is continued on the following line.
- Continue coding the value at column 16 of the following line.

Example:

```
*---+----1----+----2----+----3----+----4----+----5----+----6----+----7-
Bnn_COMMENT       = 'OUR BETA nn PRODUCTIVE SYSTEM IS NOW UP AND RUNNI*
               NG'
Bnn_TCPIP_ENCRYPT  = BFS,0102030405060708090A0B0C0D0E0F1011121314151617*
               18191A1B1C1D1E1F
```

BSA LST parameters and how to change them

Overview

This section describes all the LST parameters that are for general use.

Two types of LST parameter are listed:

- Parameters that have to be coded in member B01LST*xx*
- Global parameters that can or must be coded in member B*nn*LST*xx*

Temporary update/insert of keywords

Some LST parameters can be dynamically updated/inserted on a temporary basis using the BSA Service Manager (only for started task keywords).

For more information, see "Displaying modifiable LST parameters (Option 1.2)" in *BSA Service Manager Manual*.

SYSVAR support

Static system symbols that are activated during IPL time are permitted for all the parameters described here (see "Static system symbol support" on page 79).

LST parameters in B01LSTxx

Note The following parameters must be specified in the B01LST*xx* member. We recommend that you code them only once and use the same B01LST*xx* member for all the Beta products that are installed.

Parameter name	Value	Description	Opt./Req.	Default
SVC	*nnn*	Installation-defined SVC number used by the Beta SVC	Required	
A2G_SUPPORT	YES | NO	Enables the BQL DB cache to be used above the 2GB limit (e.g. use of main storage "above the bar") – 64 bit. If A2G_SUPPORT is active, but no storage or insufficient storage is available above the bar, the Beta product will not be started. IBM states: "*You can set a limit on how much virtual storage above the bar each address space can use. This limit is called the **MEMLIMIT**. If you do not set MEMLIMIT, the system default is 0, meaning no address space can use virtual storage above the bar. If you want to use virtual storage above the bar, you need to set the MEMLIMIT explicitly. You can set an installation default MEMLIMIT through SMFPRMxx in PARMLIB or via SETSMF command. You can also set MEMLIMIT for a specific address space in the JCL that creates the address space or by using SMF exit IEFUSI.*" Please refer to the IBM literature for further information.	Optional	NO
ARM_SUPPORT	ALL | STC | BATCH	The Automated Restart Manager of the operating system is used for the selected parameter (see "Automatic restart management (ARM) support" on page 214).	Optional	*none*
ARM_EVENT_EXIT	*name*	Name of the event exit used to prepare for the restart of an element, or to prevent the restart of an element. A sample exit in source form can be found in the BSA.SAMPLIB in member BST01AEX.	Optional	IEFBR14
SYSVAR_SUPPORT	YES | NO	Switches support for static system symbols (SYSVAR) on or off (see "Static system symbol support" on page 79)	Optional	NO

Parameter name	Value	Description	Opt./Req.	Default
DUMP_COUNTER	1..20	Number of SFF dumps to be written after an abend occurs. The maximum number is 20. If you specify a number >20, one dump will be written.	Optional	3
ESTAE	YES | NO	Switches BSA abend recovery on or off. Do not deactivate this parameter unless asked to do so by Beta Systems support. When this parameter is switched OFF, every abend will terminate the started task.	Optional	YES

LST parameters in BnnLSTxx

Parameter name	Value	Description	Opt./Req.	Default
B*nn*_SSID	ssid	The subsystem ID to be used	Required	
B*nn*_COMMENT	plaintext	Message to be written to the console when the STC is started.	Optional	*none*
IRCDE	(n,n)	Specifies the routing code to be used when generating information messages. The next table lists the valid routing codes (see "Specifying different routing codes" on page 74).	Optional	(2,0)
WRCDE	(n,n)	Specifies the routing code to be used when generating warning messages. See the following table for valid routing codes.	Optional	(12,0)
ERCDE	(n,n)	Specifies the routing code to be used when generating error messages. See the following table for valid routing codes.	Optional	(12,0)
DRCDE	(n,n)	Specifies the routing code to be used when generating decision messages. See the following table for valid routing codes.	Optional	(12,0)
DIAG	parameters	Initiates the writing of a diagnostic report to DD BSADIAG The syntax of the LST parameter DIAG corresponds to the syntax of the MODIFY command. For more information, see "Diagnostic reports" on page 295.	Optional	*none*
MSG_SUPPRESS	(msg_mask1, msg_mask2, ... msg_maskn)	This keyword and its parameters specify the message IDs or masks of message IDs of the Beta messages that are to be suppressed from output to the system log, consoles or Beta product logs. Please note that inappropriate use of this keyword can hinder debugging.	Optional	*none*

Parameter name	**Value**	**Description**	**Opt./Req.**	**Default**
DUMP_SUPPRESS	(dump_mask1, dump_mask2, ... dump_maskn)	This keyword and its parameters specify the complete abend code or masks of abend codes of the dumps which are to be suppressed from output to the SFFFDUMP and/or SYSABEND DD statement. Please note that inappropriate use of this keyword hinders the recording of a dump	Optional	*none*
MSG_ROUTE_TO	subsystem ID (ssid) of _beta access monitor	All the messages that pass the filters set in _beta access monitor are routed to the _beta access monitor subsystem *ssid*. If the keyword MSG_ROUTE_TO = *ssid* is not entered, message routing cannot be activated. If an invalid or inactive _beta access monitor subsystem ID is specified, a message will come up. Activate the _beta access monitor subsystem to start message routing. You will be notified by a message once the connection has been successfully established.	Optional	*none*
B*nn*_LICX_DSNAME	dsname	Name of the license file (required if DD B*nn*LICX is not available) For more detailed information on license files, see "License check handling" on page 18.	*see description*	*none*
B*nn*_LICX_CHECK_TIME	hh	The time as a full hour at which a cyclical check is made on licenses in the license file.	Optional	00
B*nn*_DISPLAY_LIMIT_OVERWRITE	YES | NO	When this parameter is set to YES, and the number of records specified by parameter B*nn*_DISPLAY_LIMIT_VALUE= has been reached, the user will be asked to decide whether the display limit value is to be overwritten so that more records can be read. If NO is set, the user will not be able to influence the number of records read.	Optional	NO

Parameter name	Value	Description	Opt./Req.	Default
B*nn*_DISPLAY_LIMIT_ BYPASS	YES | NO	When this parameter is set to YES, and the number of records specified by B*nn*_DISPLAY_LIMIT_VALUE= is reached for the first time, the user will be informed of this. If the user then decides to continue working, the "Limit Reached" panel will not be shown again. Instead, all the records in the table will be retrieved until the entire table can be displayed	Optional	NO
B*nn*_DISPLAY_LIMIT_ VALUE	0-999999	Specifies the number of records read before the search function stops or pauses. If B*nn*_DISPLAY_ LIMIT_BYPASS=NO, the user can continue to retrieve records until the limit value is reached again. This can be repeated as often as required. 0 means that there is no limit. All the records matching the selection criteria will be read without pausing or stopping.	Optional	1000
BSA_ANALYZE_SMPE	YES | NO	If YES, the started task or batch job outputs information on available Beta SMP/E packages and their status (see message 9350I).	Optional	NO

Note Other parameters that can be specified in BSA are described in the documentation for the facilities concerned.

Specifying different routing codes

Overview

Only the routing codes for messages written to the console can be changed. SYSLOG messages cannot be rerouted. The following table shows the values that must be entered in the parmlib member for a particular routing code.

Routing code table for Beta messages

Routing code	Definition in parmlib
1	(128,0)
2	(64,0)
3	(32,0)
4	(16,0)
5	(8,0)
6	(4,0)
7	(2,0)
8	(1,0)
9	(0,128)
10	(0,64)
11	(0,32)
12	(0,16)
13	(0,8)
14	(0,4)
15	(0,2)
16	(0,1)

Specifying more than one routing code

To specify more than one routing code for a message type, calculate the values to be entered by adding the figure to left of the comma of one value to the figure to the left of the comma of the other value, and then adding the figures to the right of the commas in the same way.

Examples

If you want to specify routing code 2 (64,0) and routing code 6 (4,0) for warning messages, add the values to the left of the comma (64 + 4 = 68) and add the values to the right of the comma (0 + 0 = 0). Enter the result (68,0) as the parameter for keyword WRCDE=:

```
WRCDE=(68,0)
```

If you want to specify routing code 3 (32,0), routing code 7 (2,0) and routing code 12 (0,16) for error messages, add the values to the left of the comma (32 + 2 + 0 = 34) and add the values to the right of the comma (0 + 0 + 16 = 16). Enter the result (34,16) as the parameter for keyword ERCDE=:

```
ERCDE=(34,16)
```

Suppressing messages

Overview

You can use the MSG_SUPPRESS parameter to suppress messages.

Suppressed messages will not appear on the console and they will not appear in logs.

Note: Message suppression does not affect the routing of messages to _beta access monitor.

Syntax

The general syntax of the message suppression statement is as follows:

```
MSG_SUPPRESS = (msg_mask1,msg_mask2,...,msg_maskn)
```

where **msg_mask*n*** is a message ID or a mask. You can use **?** and ***** as masking characters to suppress a group of messages.

Note: The **maximum length** of the suppression statement is **255 bytes**.

Examples

Following is an example of a suppression statement:

```
MSG_SUPPRESS = (OMS95*,PMS95*,???96*)
```

The statement above causes the suppression of the following messages:

- All messages beginning with **OMS95**
- All messages beginning with **PMS95**
- All messages containing **96** at positon four and five of the message ID

Continuation of long lines

Long statements can be continued on the following line like this:

- Place a continuation character at column 72 of the current line
- Continue the statement at column 16 of the following line

Following is an example of a suppression statement that continues on the following line:

```
MSG_SUPPRESS =
(QMS9501*,QMS9502*,QMS9503*,QMS9504*,QMS9505*,QMS9506*, X
               QMS9507*)
```

The statement above causes the suppression of the following messages:

- All messages (any type code) with IDs from QMS9501 through QMS9507

Suppressing dumps for SFF/RFF and SFF storage regions

Overview

When an abend occurs, a dump is produced automatically and written to the SFFFDUMP and/or SYSABEND DD statement. For some abend codes it is not necessary to produce a dump, e.g. abend codes U910, SD37, SB37, and S913. When you use dump suppression, the following occurs:

- If errors occur and the started task (STC) remains active, dumps will be suppressed for the abend codes entered.
- If errors occur and the started task (STC) ends abnormally, dumps might not be suppressed for the abend codes. This depends on the kind of the error, if e.g. the STC is seriously damaged, no dump will be suppressed.
- If a system abend and a user abend occur at the same time, then the dump will only be suppressed for the system abend and not for the user abend.

Keyword and parameters

The keyword below allows you to suppress the dumps for the specified abend codes. The general format is as follows:

```
DUMP_SUPPRESS=(dump_mask1,dump_mask2,...,dump_maskn)
```

Parameter ***dump_maskx*** contains the complete abend code, or a mask for a group of abend codes to be suppressed by the Beta system. No dump will be written to SFFFDUMP and/or to the SYSABEND DD statement for the abend codes specified here.

What to do

- Specify the abend codes you want to suppress. You can use masks.
- The codes you specify must have exactly the same format as the original abend codes. If you want to suppress user abend code U910, enter U910. The system will not suppress this user abend code if U0910 is entered, for example.

Syntax of the abend codes

The syntax of the abend codes is as follows:

S*xxx*	contains the system abend code, *xxx* stands for the abend code (up to three digits)
U*yyyy*	contains the user abend code, *yyyy* stands for the abend code (up to four digits)

The dump_mask statement can contain **?** (to signify any single character) or * (to signify a character string) as standard masking characters.

Example

For example,

```
DUMP_SUPPRESS=(U910,S?37)
```

specifies that no dump will be written when abend code U910 and abend codes such as SD37, SB37 occur.

Note

- If * is used in a mask, make sure that the asterisk is not used as the first character, but that **S** or **U** is coded before the asterisk, e.g. *S** or *U**.
- After the system has been started, you will be informed of the abend codes to be suppressed, i.e. those that will not produce a dump.
- When dump suppression takes place, the system will inform you of the situation with a message.
- The dump for abend code S722 is always suppressed. To prevent JES from producing a dump for this abend, please delete the SYSABEND DD statement in the automatically generated job or started task.

Static system symbol support

Overview

BSA supports static system symbols. Static system symbols are used to represent fixed values such as system names and sysplex names.

In the following, this function will be referred to as SYSVAR support.

Definition

Static system symbols and their substitution text are defined in member IEASYMxx in the SYS1.PARMLIB. The substitution text is defined at system initialization and remains fixed for the life of an IPL. There are two types of static system symbols:

- System-defined static system symbols, for example, &SYSCLONE, &SYSNAME, &SYSPLEX
- Installation-defined static system symbols (The system programmer specifies their names and substitution texts in the SYS1.PARMLIB dataset.)

Enabling SYSVAR support

To enable SYSVAR support, code the following keyword in member B01LSTxx in the BETA.PARMLIB:

```
SYSVAR_SUPPORT = YES
```

Coding SYSVARs

The general rules on coding system variables apply when coding SYSVARs in LST members or in the JCL. For more information on these rules, please refer to the corresponding IBM publication.

Extent of SYSVAR support

When SYSVAR support is enabled, static system variables are replaced by the substitution values defined in the IEASYMxx member. The following indicates when variables are (or are not) substituted.

- **Variables are substituted:**
  - When an STC starts:

    All static system variables that are used in the product LST members are substituted with the values defined for this system.

    Note: No variable substitution takes place when keywords defined in member B01LSTxx are processed.
  - When the datasets of a BQL database are opened:

    All static system variables that are used in the dataset names stored in the database definition file are substituted with the values defined for this system. (You cannot use static system variables in the dataset name of the database definition file itself.)
  - When a BQL database is formatted:

    All static system variables that are used in the FORMAT statement for the program BST05FOR are substituted with the values defined for this system.

- **Variables are never substituted:**
  - When the REXX tailors LST members, batch jobs, and system parameter members (for example, VTAM application ID).

    The REXX writes the names of the static system variables to the members. The variables will later be substituted by the Beta function or the JES interpreter when they need the corresponding value.
  - When BST05UPF uploads dataset names into the database definition file (DEFI file).

    The program writes the names of the static system variables to the database definition file. The variables are substituted when the corresponding datasets are opened.

Note on system symbols in dataset names

Please note the following when you are using static system symbols in dataset names of BQL databases:

Each static system symbol will be substituted with the value defined for this system. This may lead to problems when several systems share one database. You must ensure that the resulting dataset names are available on each system after variable substitution has taken place.

Note on CAF and IAF

SYSVAR support does not apply to the following:

- Parameters allocated by CAF in the CICS started task (CAFPARM)
- Parameters allocated by IAF in the IMS started task (IAFPARM)

Further reference

For more information on using and coding system symbols, see the IBM publication *MVS Initialization and Tuning Reference*.

RACF security

In this chapter

Topic	Page
User resource class $BETA	82
Security exit global parameters	84
Refreshing the RACF security user table	85

User resource class $BETA

Overview

All Beta product security checking takes place using the SAF (System Authorization Facility) calling conventions of RACF (Resource Access Control Facility), or an equivalent SAF-compatible security product. Beta product resources are defined to RACF in the user resource class $BETA.

Class descriptor table

All the resource classes used in the RACF installation are defined in the RACF class descriptor table. Classes can be defined as static classes or as dynamic classes.

BSA.SAMPLIB members for defining $BETA

Member RACF#CDT in the BSA.SAMPLIB contains a set of sample RACF commands for defining $BETA as a static class or as a dynamic class.

Modify the sample members to suit your security requirements. Parameters ID= (static classes only) and POSIT= (static and dynamic classes) must be modified so that they are unique at your installation.

Member RACF#ASM in the BSA.SAMPLIB contains a sample job to assemble and link the class descriptor table.

Static or dynamic class

Defining a static class requires an IPL to enable this class to take effect.

Defining a dynamic class does not require an IPL.

IBM recommendation: All classes that are not standard IBM classes should be defined dynamically.

Defining a static class

Following is an example for defining $BETA as a static class:

```
+----------------------------------------------------------------------+
|        PRINT GEN                                                     |
|$BETA   ICHERCDE CLASS=$BETA,                                      +  |
|              ID=145,                                              +  |
|              MAXLNTH=64,                                          +  |
|              FIRST=ANY,                                           +  |
|              OTHER=ANY,                                           +  |
|              POSIT=45,                                            +  |
|              OPER=NO,                                             +  |
|              DFTUACC=NONE                                            |
|        ICHERCDE                                                      |
|        END                                                           |
+----------------------------------------------------------------------+
```

Note: An IPL is necessary for this newly defined class to take effect.

Defining a dynamic class

Following is an example for defining $BETA as a dynamic class:

```
+-------------------------------------------------------------------+
| RDEFINE  CDT  $BETA  CDTINFO(                                   + |
|                               MAXLENGTH(64)                     + |
|                               FIRST(ALPHA,NATIONAL,NUMERIC,SPECIAL) + |
|                               OTHER(ALPHA,NATIONAL,NUMERIC,SPECIAL) + |
|                               POSIT(45)                         + |
|                               OPERATIONS(NO)                    + |
|                               DEFAULTUACC(NONE)                 + |
|                             )                                     |
+-------------------------------------------------------------------+
```

Note: `SETROPTS RACLIST (CDT) REFRESH` must be executed for this newly defined class to take effect.

Sample definitions

The samples are defined as follows:

- **MAXLNTH / MAXLENGTH**:The maximum length of a resource name is 64 characters. This is sufficient for all Beta product resources.
- **FIRST** and **OTHER**: ANY characters can be used in both the FIRST and OTHER position of the resource names of static classes. When working with dynamic classes, however, FIRST and OTHER must be specified as shown in the second example above.

  Note: The set of characters that can be used is restricted by the Beta product security exit B*nn*UXSEC, which allows only a subset of special characters for resource names. See the description of the sample product security exit in the product *Installation and System Guide* for a list of permitted characters.
- **OPER / OPERATIONS**: A user with the attribute OPERATIONS does not automatically have access to this class.
- **DFTUACC /DEFAULTUACC**: The default universal access is NONE. The default universal access comes into effect whenever a profile is defined to RACF without specifying the universal access (UACC) for this resource.

Notes

- We recommend that you specify parameter `RACLIST=ALLOWED` to optimize the use of resources.
- If you want to define generic profiles in class $BETA, remember to define them as generic and to activate generic profile checking (options GENCMD and GENERIC). This has to be done before the first profile is defined.
- For more information, see the IBM manual *RACF Macros and Interfaces*.

Security exit global parameters

Overview

Each Beta product calls its own security exit (B*nn*UXSEC) to protect access to resources within the product. The security exit is delivered in source form in the BETA*nn*.SAMPLIB.

Parameter lists

The security exit program points to two parameter lists:

- A global section containing the parameters that are used for all products. The DSECT for these parameters is supplied in member B00#SSEC of the BSA.SAMPLIB. It is described below.
- A product specific section containing parameters used for a specific product only. A DSECT for these parameters is delivered in the BETA*nn*.SAMPLIB (member B*nn*#SSEC) and is described in the _beta product *Installation and System Guide.*

Global parameter list for B*nn*UXSEC (B00#SSEC)

Displacement	Description
+24	Subsystem ID
+28	Calling program name
+36	User ID
+44	Line command
+48	SMF ID
+52	First letter of access type
+53	Beta product ID (e.g. 93)
+55	If not 0, pointer to B*nn*#SSEC
+59	Product security call ID (function code)

B*nn*#SSEC parameters

The B*nn*#SSEC parameters are described in the _beta product *Installation and System Guide*.

Refreshing the RACF security user table

Overview

RACF user information is contained in the RACF security user table of the Beta product. Beta Query Language (BQL) keeps the RACF security user table in storage. This storage region is used to hold data temporarily. If a RACF user profile is added to the RACF security user table while the system is running, the table will need to be refreshed. There are three ways of doing this:

- Logging off and restarting the STC
- Using the Refresh command at the console
- Using option **2.3** of the BSA Service Manager

Refresh command at the operator console

To refresh the RACF security user table, execute a MODIFY command for the started task (STC). The syntax of the operator console command is as follows:

```
F  stcname,REFRESH RCF[,U=userid]
```

-OR-

```
F  stcname,REFR RCF[,U=userid]
```

Masks cannot be used in this command.

Result

- When a user ID is specified, the Refresh will be executed for that user. If not, the Refresh will be executed on the entire table.
- When the command is issued, you will receive a message that the Refresh command is active. The next time a BQL command is executed, the RACF security user table will also be refreshed. The user does not have to relogon.

Changing the number of users

The default for the RACF security user table keyword is:

```
BQL_USER = 500
```

Change this value to increase or decrease the number of users allowed in the RACF security user table, and then execute the Refresh command.

Switching off the RACF security user table

If you do not need the RACF user information in storage, specify the following keyword to switch off the RACF security user table:

```
BQL_USER = 0
```

Multi-CPU with the Open Communication Facility (OCF)

In this chapter

Topic **Page**

Introduction 87
Overview of OCF functions and parameters 88
OCF environment 89
Configuring OCF for a multi-CPU environment 91
Customization 92
VTAM resource definitions 93
OCF network resource definitions 94
LST parameters for OCF 95
Calculating the values for RU sizes 98
Security in a host-to-host multi-CPU environment 100
OCF operator console commands 101
Examples of OCF configurations 102
  Example 1: Host-to-host - 4 STCs and 4 OCF nodes (recommended) 103
  Example 2: Host-to-host - 4 STCS and 2 OCF nodes 105
Establishing OCF communication via X-System Router (Beta 02) 107
  Example 1: X-System Router Beta 02 with slave system 108
  Example 2: X-System Router Beta 02 without a slave system 109
  Parameters for OCF via TCP/IP 110

Introduction

What is the Open Communication Facility?

The Open Communication Facility (OCF) is an accessibility tool providing cross-system access to Beta product functions across multiple CPU platforms.

The Open Communication Facility enables quick and reliable access to data center information in a multi-CPU configuration.

Protocols

OCF provides cross-product communication between Beta products on different LPARs using the Advanced Program-to-Program Communication (APPC/VTAM) implementation of the SNA LU6.2 communications protocol. Optionally, OCF allows the APPC interface to be emulated across an LU2 VTAM link.

Alternatively, you can also use OCF via TCP/IP if you are using the BSA X-System Router Beta 02 (see "Parameters for OCF via TCP/IP" on page 110).

When to use

Use OCF if you want to provide a communications link between Beta products that are on different sysplexes.

Use XCF instead if you want to provide a communications link between Beta products that are on different LPARs within one sysplex (see "Multi-CPU using BSA XCF in a sysplex" on page 116).

Who should read this section?

This section is intended for the systems programmer responsible for installing and maintaining the Open Communication Facility functions.

Overview of OCF functions and parameters

OCF local functions

OCF local functions are subsystem functions providing cross-system communications services for Beta product started tasks, batch jobs, online users, or programs running on the same operating system as the OCF local functions. Depending on the Beta product and the product features, OCF functions can either run exclusively in their own address space, or they can run in the same address space as the Beta product started tasks.

OCF remote functions

OCF remote functions are subsystem functions providing cross-system communication services for Beta product started tasks, batch jobs, online users, or programs running on an operating system connected via an SNA network to the operating system running the local OCF functions. Again, depending on the Beta product implementation and product features, the OCF remote functions can either be running exclusively in their own address space or they can be running in the same address space as the Beta product started tasks.

OCF parameters

During the installation of any Beta product requiring the services of the Open Communication Facility, a set of startup parameters is created. Parameters are tailored to your installation requirements which you specify when running the installation REXX. When you submit the installation jobs, the startup parameter members are placed into the BETA.PARMLIB member assigned to the Beta product concerned.

The OCF startup parameters are used for controlling how VTAM resources are used when local subsystems and batch jobs communicate with remote subsystems. Please refer to the description of the startup deck (see "OCF network resource definitions" on page 94).

OCF environment

Overview

The Open Communication Facility is a set of programs that make use of VTAM Advanced Program-to-Program Communication (APPC/VTAM) and Beta Subsystem Function Facility (SFF) services to enable the efficient transfer of information between application subsystems residing on the same computing systems on different LPARs, or on different computing systems. These application subsystems may either be running on z/OS as a Beta product subsystem, or on an end-user workstation as a Beta product server application or end-user presentation interface.

Subsystem Function Facility (SFF) services

OCF uses the Beta Subsystem Function Facility services for task dispatching, storage control, program control and serialization of OCF functions. Among the benefits of SFF services are the efficient startup, termination and dispatching of work to and from the OCF and application service transactions executed in the course of information transfer.

How OCF uses APPC

OCF uses the VTAM implementation of Advanced Program-to-Program Communication services to provide communication between application service programs. APPC/VTAM is the VTAM implementation of a subset of the Systems Network Architecture LU6.2 communications protocol, which is also used by a variety of other hardware and software manufacturers.

Benefits of APPC services

Among the benefits of APPC services are the protection of application programs from changes to the communications hardware environment, independence from the type of processor hardware being communicated with, automatic service transaction startup and termination, transmission buffering and program synchronization services.

How OCF complements APPC

With OCF and APPC, the application service programs can concentrate on executing and distributing the actual application functions across multiple CPU platforms without having to consider how, to where or on what type of connection the information transfer takes place. This type of environment is called a distributed transaction processing environment, or, more commonly, a client-server environment.

OCF complements APPC by providing additional services that enable more efficient use of APPC resources when transferring information between Beta product functions. In this respect, OCF is the window to communications services for Beta product functions.

How OCF nodes work

Each OCF node in the OCF network exchanges information with other OCF nodes about the Beta product type and subsystem identification of Beta product address spaces running on the local z/OS. This information is contained in the BETA.PARMLIB member B*nn*LST*xx*, which is processed during subsystem startup by each OCF node started task. In this way, remote OCF nodes automatically know where to route work requests if the target subsystem does not reside on the local z/OS system.

Consequently, the subsystem identification (SSID) names of the subsystems communicating through an OCF network must be unique. Furthermore, each OCF node can be a member of only one OCF network.

Details of OCF environments

In the following instructions we describe specific environments and Beta functions in which the OCF is implemented. For definition purposes, we will refer to the set of OCF functions that are logically connected to each other via a VTAM network as the "OCF Network".

Configuring OCF for a multi-CPU environment

General requirement

On each operating system running Beta product address spaces, there must be at least one address space (the "OCF node") for OCF functions, enabling the Beta products to communicate with each other across the OCF network. At present, OCF node functions always run in the started task of a product or enhancement facility.

Note: Each OCF node can be a member of only one OCF network.

Master-slave or master-master?

As in the examples, the product subsystems are best connected as master and slave (master-slave connection). Depending on the features of the Beta product concerned, a master-to-master connection can be set up, but only for online usage (TSO or Beta VDF).

Product-specific examples

Each Beta product may have slightly different requirements. For detailed examples of OCF environments for a specific product, please see the *Installation and System Guide* of the product concerned.

BETA.PARMLIB definitions

To support the VTAM connection between the two z/OS systems, some definitions must be made in the SYS1.VTAMLST (see "VTAM resource definitions" on page 93) and in the BETA.PARMLIB datasets of the OCF node subsystems (see "OCF network resource definitions" on page 94). Once in place, these definitions need not be changed again unless changes are made to the LU6.2 connection, or additional Beta product subsystems are added to the OCF Network.

Customization

Overview

In order to describe the environment and resources used in an OCF network, two sets of definitions must be made tailored to your installation requirements. Additional steps must be performed to secure online access to remote Beta product subsystems.

VTAM resource definitions

VTAM resource definitions (see "VTAM resource definitions" on page 93) define the VTAM resources that will be used by the OCF network nodes.

OCF network definitions

These describe to the OCF functions the names and types of Beta product subsystems residing on the local z/OS system which are addressable via the OCF network. The OCF network definitions (see "OCF network resource definitions" on page 94) also describe the names of the partner OCF nodes with which the local OCF node can converse.

Initialization

Finally, you must perform the same initialization steps on the local z/OS system, with the same subsystem ID, security exits, and security profiles as on the remote CPU. This is done to provide the same level of security checking on the local z/OS system as would be required on the remote z/OS system to access the resources in the Beta product started task.

Changes

Once in place, these definitions need not be reconsidered unless changes are made to the VTAM environment, or Beta product subsystems are added to the OCF network.

VTAM resource definitions

Overview

The SYS1.VTAMLST dataset contains the VTAM definitions required to define the OCF node to the VTAM network. Each OCF node identifies itself to the VTAM network via the OCF_APPLID keyword found in the corresponding B*nn*LST*xx* member of the BETA.PARMLIB. This name is the VTAM application ID of the OCF node.

APPC application definition

The following shows the general structure of each VTAM application definition:

```
+-------------------------------------------------------------------------+
|application_id APPL AUTH=(ACQ,SPO),APPC=YES,MODETAB=table_name,AUTOSES=s,|
|                    SECACPT=CONV,DMINWNL=n1,DMINWNR=n2,DSESLIM=n3,       |
|                    DRESPL=ALLOW                                         |
+-------------------------------------------------------------------------+
```

When you specify the use of OCF in a Beta product installation procedure, a VTAM application definition is generated in the Beta product CNTL dataset and copied to SYS1.VTAMLST by an installation batch job.

Log mode

The following logmode must be defined for a host-to-host-connection. The name APPC01 is the default. The following shows a valid MODETAB definition (see ***table_name*** above):

```
+-------------------------------------------------------------------------+
|MODEAPPC MODETAB                                                         |
|APPC01   MODEENT LOGMODE=APPC01,                                       X |
|               FMPROF=X'13',                                           X |
|               TSPROF=X'07',                                           X |
|               PRIPROT=X'B0',                                          X |
|               SECPROT=X'B0',                                          X |
|               COMPROT=X'D0B1',                                        X |
|               RUSIZES=X'F7F7',                                        X |
|               PSERVIC=X'060200000000000000002000',                    X |
|               PSNDPAC=X'03',SRCVPAC=X'03',SSNDPAC=X'00|'                |
|SNASVCMG MODEENT LOGMODE=SNASVCMG,FMPROF=X'13',TSPROF=X'07',           X |
|               PRIPROT=X'B0',SECPROT=X'B0',COMPROT=X'D0B1',            X |
|               RUSIZES=X'8585',ENCR=B'0000',                           X |
|               PSERVIC=X'060200000000000000000300'          *R495812*   |
|*                                                                        |
|*                                                                        |
|         MODEEND                                                         |
+-------------------------------------------------------------------------+
```

Note

If the local and remote LUs are to reside on different VTAM subareas, the remote LU must also be specified locally as a VTAM cross-domain resource (CDRSC) with the MODETAB parameter specifying the above MODETAB table.

A sample is placed into member MODEAPPC of the BSA.SAMPLIB and can be assembled by ASMMODE, which is also a member in the BSA.SAMPLIB.

OCF network resource definitions

Overview

The BETA.PARMLIB contains a startup deck of information used by each OCF node to determine which network of OCF nodes to connect to, and the names and types of Beta product subsystems that are addressable via this started task.

Any changes to the OCF network definitions in the parmlib dataset (and subsequent changes to the OCF network topology) only become active after the OCF node subsystem has been stopped and restarted.

OCF network resource definition

The following shows the general structure of each OCF network resource definition:

```
+-------------------------------------------------------------------------+
|OCF_APPLID       = local_applid                                          |
|OCF_CONVERSE     = 'remote_applid_1,sessions_1,retries_1,interval_1,maxdata_1' |
|OCF_CONVERSE     = 'remote_applid_2,sessions_2,retries_2,interval_2,maxdata_2' |
|                    .                                                    |
|                    .                                                    |
|                    .                                                    |
|OCF_CONVERSE     = 'remote_applid_N,sessions_N,retries_N,interval_N'     |
|OCF_LOCAL_SYSTEM = 'subsysid_1,product_1'                                |
|OCF_LOCAL_SYSTEM = 'subsysid_2,product_2'                                |
|                    .                                                    |
|                    .                                                    |
|                    .                                                    |
|OCF_LOCAL_SYSTEM = 'subsysid_M,product_M'                                |
+-------------------------------------------------------------------------+
```

The following table describes each parameter in detail.

Static system symbols that are activated at IPL time are allowed for all the parameters described here (see "Static system symbol support" on page 79).

LST parameters for OCF

Overview

All the parameters listed in this table are automatically inserted into the B*nn*LST*xx* member of the Beta product when you specify during installation that you want to use the product with OCF.

Keyword	**Value**	**Description**	**Option**	**Default**
OCF_APPLID	vtamapplid	The VTAM application ID of this OCF node. The application name must be defined in the SYS1.VTAMLST dataset (see "VTAM resource definitions" on page 93) with parameter APPC=YES. **Note**: The SSID used must be unique.	required	Depends on the Beta product serving as the OCF node.
OCF_CONVERSE	*remapplid*, *sessions*, *retries*,*interval*, *maxdata*	Details of the five positional parameters of this keyword are provided below; remapplid is required; the other parameters are optional		
	remapplid	The VTAM application IDs of the OCF nodes with which the OCF node specified by OCF_APPLID is connected. The number of remote OCF nodes with which this node can converse is mainly limited by the amount of VTAM resources available to the installation. As with the keyword OCF_APPLID, these application names must be defined in the SYS1.VTAMLST (see "VTAM resource definitions" on page 93) dataset with parameter APPC=YES. This keyword also sets some limits on how the node-to-node conversations can take place (see five optional parameters below). **Note**: The SSID used must be unique.	required	none
	sessions	The maximum number of concurrent LU-LU sessions that can take place at one time with this remote OCF node. The value specified here is limited by the session limits set in the application definition in SYS1.VTAMLST. If the value specified there is lower than this one, the value specified in SYS1.VTAMLST will be used.	optional	4

Keyword	Value	Description	Option	Default
	retries	The number of times a BIND attempt is retried before OCF determines that a physical connection to the remote node is not possible. After this, OCF waits for a BIND request from the partner (specified in 'remapplid').	optional	5
	interval	The minimum number of seconds that OCF is to wait before retrying a failed BIND attempt.	optional	10
	maxdata	The maximum size of a Path Information Unit (PIU) that can be transferred between OCF nodes. The maximum permitted size depends on the VTAM connection, e.g. CTC attached, VTAM-NCP attached, etc. If the value specified here exceeds the permitted size, it will not be possible to correctly establish a connection between the OCF nodes. Message 9241E or 9243E will come up. See the explanation of message 9204E in *BSA Messages and Codes* to find out how to determine the size used for MAXDATA. If a value for MAXDATA is not entered here, or if the value is 0 (default), the RU size from the LOGMODE APPC01 of the MODETAB definition will be used instead (see "Calculating the values for RU sizes" on page 98).	optional	0
OCF_ACTSESS_ MOD	YES | NO	Activates the MAXDATA or RU size values as entered in the keyword OCF_CONVERSE. **YES:** The value entered for MAXDATA in the keyword OCF_CONVERSE or the RUSIZE value entered in the LOGMODE APPC01 of the MODETAB definition will be activated. Message 9231I informs you which parameter is active. **NO:** The value entered for MAXDATA in the keyword OCF_CONVERSE or the RUSIZE value entered in the LOGMODE APPC01 of the MODETAB definition will not be used. Message 9231I informs you which parameter is active.	optional	NO

Keyword	Value	Description	Option	Default
OCF_LOCAL_ SYSTEM	*ssid*,*product*	Both positional parameters are required.		
	ssid	The subsystem ID of the Beta product started task which is addressable via this OCF node.	required	none
	product	The Beta product running under the started task.	required	none
		The subsystem IDs and Beta product types of the Beta product subsystems that are executing on the same operating system as this OCF node. The subsystems specified here are the target subsystems for communications and data transfer from other subsystems connected to this OCF network. Furthermore the subsystems specified here can send requests to remote subsystems connected to the remote OCF nodes which are specified in the OCF_CONVERSE parameter. The number of local subsystems that can be connected to this OCF node is mainly limited by the amount of Common Service Area (CSA) available to the installation.		
OCF_LOGMODE_ NAME	name	Name of the LOGMODE used in the VTAM MODETAB.	optional	APPC01

Calculating the values for RU sizes

Calculating the RU size

When the MAXDATA value is known, the RU size (Request Unit size) is calculated as follows: MAXDATA value minus 34 bytes (header of PIU) equals the RU size, e.g. 2048 - 34 = 2014 bytes. (See the explanation of message 9204E in manual *BSA Messages and Codes* on how to determine the MAXDATA size.) If no matching equal value is returned in the table, e.g. 2014, then the next lower value must be used for the maximum permitted RU size, e.g. 1920, displayed as a hexadecimal X'F7F7'.

Formula

The formula for calculating the RU size value is: a x 2b or X'abab', displayed as a hexadecimal.

The formula specifies the maximum length in bytes of the RUs (request units) that the primary and the secondary logical unit can send to each other. The RU sizes are specified as four hexadecimal figures: X'abab'. The first digit is the mantissa (a) and must be within the range X'8'-X'F'. The second digit is the exponent (b) and must be within the range X'0'-X'F'. The leftmost two digits apply to the secondary logical unit and the rightmost two digits apply to the primary logical unit. These sent and received units must always be of the same size.
X'F7F7' specifies that the maximum size of the unit sent or received is 15 x 27 or 1920 bytes.

Table of exponents and mantissas

The table below lists the exponents, mantissas, and the values for RU sizes.

	Mantissa (a)							
Exponent (b)	**8**	**9**	**A (10)**	**B (11)**	**C (12)**	**D (13)**	**E (14)**	**F (15)**
0	8	9	10	11	12	13	14	15
1	16	18	20	22	24	26	28	30
2	32	36	40	44	48	52	56	60
3	64	72	80	88	96	104	112	120
4	128	144	160	176	192	208	224	240
5	256	288	320	352	384	416	448	480
6	512	576	640	704	768	832	896	960
7	1024	1152	1280	1408	1536	1664	1792	1920
8	2048	2304	2560	2816	3072	3328	3584	3840
9	4096	4608	5120	5632	6144	6656	7168	7680
A (10)	8192	9216	10240	11264	12288	13312	14336	15360
B (11)	16384	18432	20480	22528	24576	26624	28672	30720
C (12)	32768	36864	40960	45056	49152	53248	57344	61440
D (13)	65536	73728	81920	90112	98304	106496	114688	122880
E (14)	131072	147456	163840	180224	196608	212992	229376	245760
F (15)	262144	294912	327680	360448	393216	425984	458752	491520

Security in a host-to-host multi-CPU environment

Overview

If you intend to allow online users access to Beta product subsystems on remote CPUs, you must initialize the remote security environment on the local CPU.

To do this, you must perform the same Beta product subsystem initialization (via the program BST01ARI, during IPL or in batch) on the local CPU as on the remote CPU where the Beta product subsystem actually resides.

Definitions

For example, if your local OCF node subsystem ID is **B09P**, and the remote Beta product subsystem ID your local online users require access to is **B92P**, then the following IEFSSNxx definitions must be accessible to the local z/OS system:

```
+----------------------------------------------------------------------+
|B09P,BST01ARI,'BETA.PARMLIB(B09SSI00)'                                |
|B92P,BST01ARI,'BETA.PARMLIB(B92SSI00)'                                |
+----------------------------------------------------------------------+
```

You can also define and initialize the subsystems dynamically. For each subsystem, enter the command `SETSSI ADD,SUB=`*`ssid`* and then run B09INIT/B92INIT on the local CPU.

SVC number

In addition, the SVC number specified in B92SSI00 (using the SVC= parameter) must be initialized on the local z/OS system. If the Beta SVC number in use on the remote system is different from the one in use on the local system, you must change B92SSI00 (by copying it to a new member such as B92SSI01) so that it contains the Beta SVC number in use on the local system, and use the following IEFSSNxx entries instead:

```
+----------------------------------------------------------------------+
|B09P,BST01ARI,'BETA.PARMLIB(B09SSI00)'                                |
|B92P,BST01ARI,'BETA.PARMLIB(B92SSI01)'                                |
+----------------------------------------------------------------------+
```

Security exits

Finally, the security exits and security environment used for online users working on the remote CPU must also be available on the local CPU. This is because security checking (logon and access validation) is always performed on the local CPU, even if access is to a remote Beta product subsystem.

For more information on defining the Beta product subsystems and security environments to z/OS, please refer to the appropriate Beta product *Installation and System Guide*.

OCF operator console commands

Display OCF connections

This command displays all OCF connections supported by a started task:

```
F betastc,D OCF[,ALL]
```

where *betastc* is the Beta started task (STC).

All relevant information, such as the supported subsystem IDs, their status and the general status of the connections is displayed at the operator console.

ALL is optional. When ALL is entered, all relevant data concerning the connection control of the control blocks in use is also displayed. In this case, the data is displayed in the SYSLOG.

Messages 9221I and 9222I are written to the console.

Examples of OCF configurations

OCF host-to-host configurations

In the following, examples 1 and 2 show host-to-host configurations. The first example shows the recommended configuration. Both examples establish OCF communication within the started task of a product.

Both examples show two online (TSO or Beta VDF) users who can access the subsystems of a specific Beta product (e.g. B93A/ B93P and B92A/B92B) in the OCF network. The recommended configuration between subsystems of the same product is a master-slave connection.

Key to diagrams

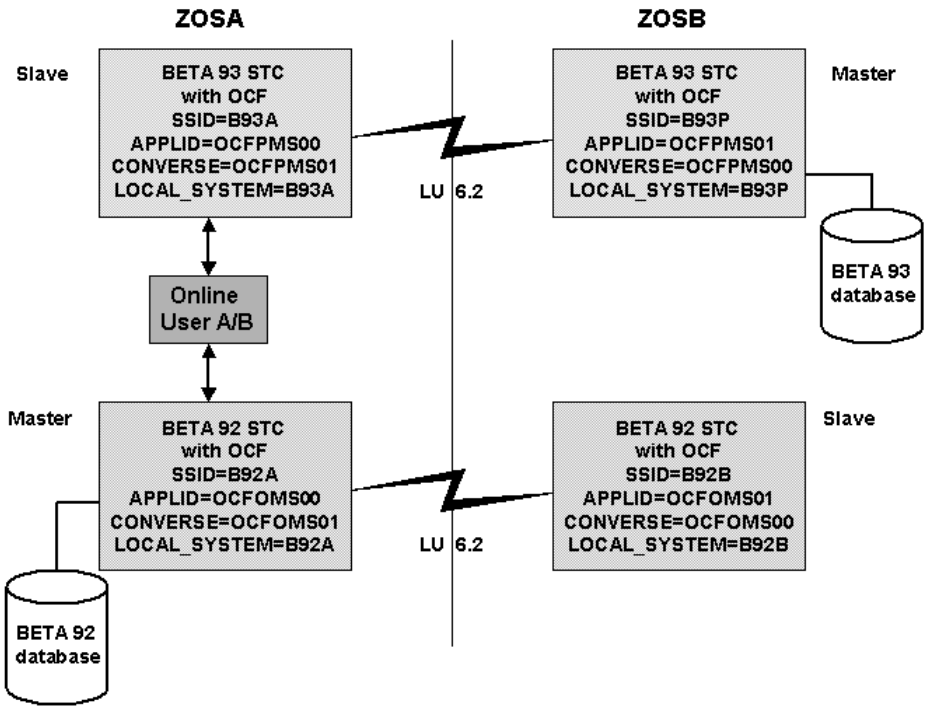
APPL=	VTAM Application ID of the OCF node
APPLID=	VTAM Application ID of the OCF node
BETA BOF STC	Base Output Facility started task
Beta SVC	Beta SVC used for data transfer between subsystems in one operating system
BETA93 STC	Beta 93 started task
BETA9x STC	Beta 92, Beta 93 started task
BETA9x STC or JOB	Beta 92, Beta 93 product address space (started task or batch job)
OCF	Open Communication Facility
Online user A/B	TSO user or Beta VAF/CAF/IAF address space
SSID=	Subsystem ID of the started task

Example 1: Host-to-host - 4 STCs and 4 OCF nodes (recommended)

Overview

In example 1, four address spaces are shown conversing with each other in an OCF network. Two of the address spaces, subsystems B93A and B92A, are on system ZOSA. The other two, B93P and B92B, are on system ZOSB. In addition to carrying out Beta product application functions, all of the subsystems serve as OCF nodes transferring requests to the other machine via the LU6.2 protocol boundary. Because they are being used as OCF nodes, all four subsystems must remain active for the entire period of communication between ZOSA and ZOSB.

Diagram example 1


ZOSA
ZOSB
Slave
BETA 93 STC with OCF SSID=B93A APPLID=OCFPMS00 CONVERSE=OCFPMS01 LOCAL_SYSTEM=B93A
BETA 93 STC with OCF SSID=B93P APPLID=OCFPMS01 CONVERSE=OCFPMS00 LOCAL_SYSTEM=B93P
Master
LU 6.2
BETA 93 database
Online User A/B
Master
BETA 92 STC with OCF SSID=B92A APPLID=OCFOMS00 CONVERSE=OCFOMS01 LOCAL_SYSTEM=B92A
BETA 92 STC with OCF SSID=B92B APPLID=OCFOMS01 CONVERSE=OCFOMS00 LOCAL_SYSTEM=B92B
Slave
LU 6.2
BETA 92 database

How the subsystems communicate

Subsystem B92A on system ZOSA can communicate with subsystem B92B in system ZOSB and vice versa simply by specifying the subsystem ID B92B in the transfer request. The request is then automatically routed from subsystem B92A to the target subsystem B92B. Likewise, subsystem B93A on system ZOSA can communicate with subsystem B93P in system ZOSB.

User access

Online users A and B on system ZOSA can access subsystem B93P (master) on ZOSB simply by specifying the subsystem (B93A, slave, or B93P directly) in the online application. Likewise, online users A and B on system ZOSA can access subsystem B92B (slave) on ZOSB simply by specifying the subsystem (B92A, master) in the online application. The online user only needs to know the system and location name, and the subsystem ID. OCF takes care of the rest.

Definition as OCF nodes

The subsystems of each Beta product are separately defined as OCF nodes, e.g. there is a direct connection between the subsystems B92A and B92B, and a direct connection between B93A and B93P.

BETA.PARMLIB definitions

In this example, the following OCF parameters are necessary in the BETA.PARMLIB members:

- In the active B93LST*xx* member on ZOSA:

```
+--------------------------------------------------------------------+
|     ...                                                            |
|OCF_APPLID       =   OCFPMS00                                       |
|OCF_CONVERSE     =   OCFPMS01,4,5,10,32768                          |
|OCF_LOCAL_SYSTEM =   B93A,BETA93                                    |
+--------------------------------------------------------------------+
```

- In the active B92LST*xx* member on ZOSA:

```
+--------------------------------------------------------------------+
|     ...                                                            |
|OCF_APPLID       =   OCFOMS00                                       |
|OCF_CONVERSE     =   OCFOMS01,4,5,10,32768                          |
|OCF_LOCAL_SYSTEM =   B92A,BETA92                                    |
+--------------------------------------------------------------------+
```

- In the active B93LST*xx* member on ZOSB:

```
+--------------------------------------------------------------------+
|     ...                                                            |
|OCF_APPLID       =  OCFPMS01                                        |
|OCF_CONVERSE     =  OCFPMS00,4,5,10,32768                           |
|OCF_LOCAL_SYSTEM =  B93P,BETA93                                     |
+--------------------------------------------------------------------+
```

- In the active B92LST*xx* member on ZOSB:

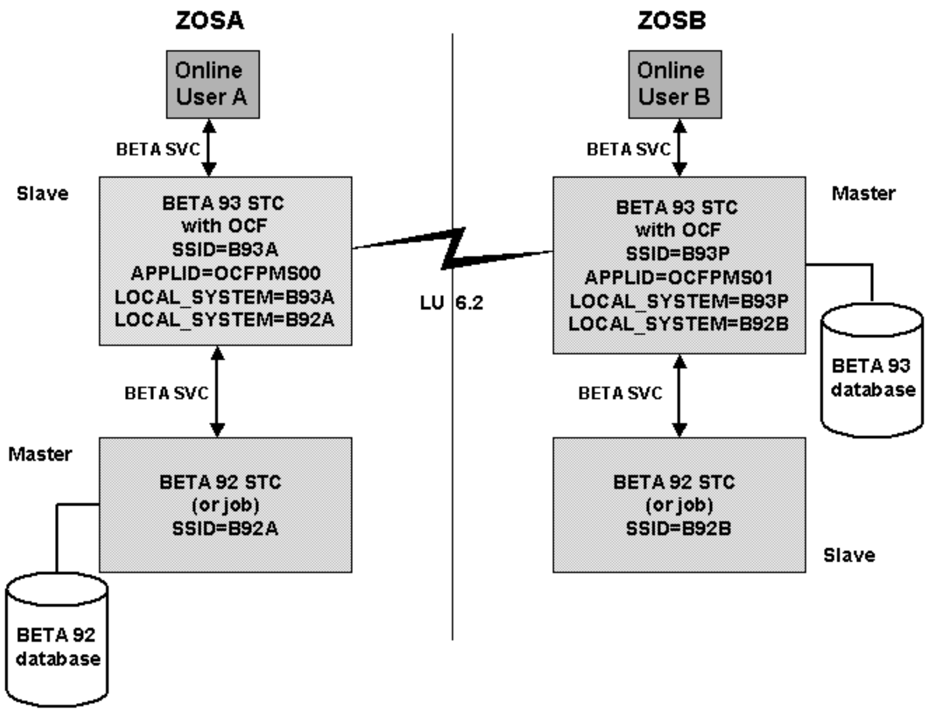
```
+--------------------------------------------------------------------+
|     ...                                                            |
|OCF_APPLID       =  OCFOMS01                                        |
|OCF_CONVERSE     =  OCFOMS00,4,5,10,32768                           |
|OCF_LOCAL_SYSTEM =  B92B,BETA92                                     |
+--------------------------------------------------------------------+
```

Example 2: Host-to-host - 4 STCS and 2 OCF nodes

Overview

In example 2, four address spaces are shown conversing with each other in an OCF network. Two of the address spaces, B93A and B92A, are on system ZOSA. The other two, B93P and B92B, are on system ZOSB. In addition to carrying out Beta product application functions, subsystems B93A and B93P also serve as OCF nodes in transferring requests to the other machine via the LU6.2 protocol boundary. Because they are being used as OCF nodes, both subsystems (B93A and B93P) must remain active for the entire period of communication.

Diagram example 2


ZOSA
ZOSB
Online User A
Online User B
BETA SVC
BETA SVC
Slave
BETA 93 STC with OCF SSID=B93A APPLID=OCFPMS00 LOCAL_SYSTEM=B93A LOCAL_SYSTEM=B92A
BETA 93 STC with OCF SSID=B93P APPLID=OCFPMS01 LOCAL_SYSTEM=B93P LOCAL_SYSTEM=B92B
Master
LU 6.2
BETA 93 database
BETA SVC
BETA SVC
Master
BETA 92 STC (or job) SSID=B92A
BETA 92 STC (or job) SSID=B92B
Slave
BETA 92 database

How the subsystems communicate

Subsystem B92A on system ZOSA can communicate with subsystem B92B in system ZOSB and vice versa simply by specifying the subsystem ID B92B in the transfer request. The request is then automatically routed from subsystem B92A via B93A and B93P to the target subsystem B92B.

User access

Online user A on system ZOSA can access subsystem B93P (master) on ZOSB simply by specifying the subsystem (B93A, slave) in the online application. The online user only needs to know the system and location name, and the subsystem ID. OCF takes care of the rest. The same applies to online user B. User B has direct access to subsystem B93P (master).

BETA.PARMLIB definitions

In this example, the following OCF parameters are necessary in the BETA.PARMLIB members:

- In the active B93LST*xx* member on ZOSA:

```
+------------------------------------------------------------------------+
|     ...                                                                |
|OCF_APPLID       =   OCFPMS00                                           |
|OCF_CONVERSE     =   OCFPMS01,4,5,10,32768                              |
|OCF_LOCAL_SYSTEM =   B93A,BETA93                                        |
|OCF_LOCAL_SYSTEM =   B92A,BETA92                                        |
+------------------------------------------------------------------------+
```

- In the active B93LST*xx* member on ZOSB:

```
+------------------------------------------------------------------------+
|     ...                                                                |
|OCF_APPLID       =  OCFPMS01                                            |
|OCF_CONVERSE     =  OCFPMS00,4,5,10,32768                               |
|OCF_LOCAL_SYSTEM =  B93P,BETA93                                         |
|OCF_LOCAL_SYSTEM =  B92B,BETA92                                         |
+------------------------------------------------------------------------+
```

For the BETA.PARMLIB members in use by Beta 92 on ZOSA and Beta 92 on ZOSB, no additional OCF parameters are required. This is because the subsystems B92A and B92B have already been specified in the OCF_LOCAL_SYSTEM parameters above as being accessible in this OCF network.

Alternative configuration

It would also be possible to connect subsystems B92A and B93P directly across the LU6.2 boundary, in addition to the connection between the subsystems B93A and B93P. As a consequence, it would be necessary to maintain additional VTAM and OCF definitions and additional hardware to support the additional conversations. Since the OCF nodes on each z/OS machine know which subsystems are locally active, only one OCF and VTAM connection is required between two z/OS hosts running Beta products. The hardware and software resources required are then put to use more efficiently.

Establishing OCF communication via X-System Router (Beta 02)

Overview

OCF communication can also be established using the BSA X-System Router Beta 02. As a separate standalone started task, the X-System Router enables two LPARS (operating systems) to be connected via OCF. It can also be used to connect two parallel sysplexes, or to connect a parallel sysplex to an external system.

Advantages of using Beta 02

The advantage of using the X-System Router for OCF communication are as follows:

- This started task does not require a database or a Beta product library.
- In the event of VTAM problems, it can be restarted to rebuild the OCF (for example Beta 93) that is active as the master.
- Beta 02 can be used as an OCF node in a multi-CPU XCF connection without influencing the functions of the product started task environment (parallel sysplex) to provide a communication link to an external system.

Installation procedure

The X-System Router is implemented by installing Beta 02 as a started task. It is installed according to the same rules as for all other Beta started tasks. Do the following:

1. Create the JCL for the Beta 02 started task.
2. Create member B02SSIxx for the initialization of the subsystem ID (see "SSI parameters" on page 145).
3. Define the subsystem ID.
4. Run BST01ARI.
5. Customize LST member B02LSTxx with the required OCF parameters (see "LST parameters for OCF" on page 95 and "Parameters for OCF via TCP/IP" on page 110.

Note

X-System-Router Beta 02 can also be used as a standalone started task for the BSA TCP/IP server. An example of how to activate the server as a separate, standalone, Beta 02 started task can be found under the name BETA02 in the BSA.SAMPLIB. For more information, see "BSA TCP/IP server as a standalone STC (Beta 02)" on page 144.

Examples

Two examples are given on the following pages. The first example shows how the X-System Router can be used with a slave system, the second shows it without a slave system.

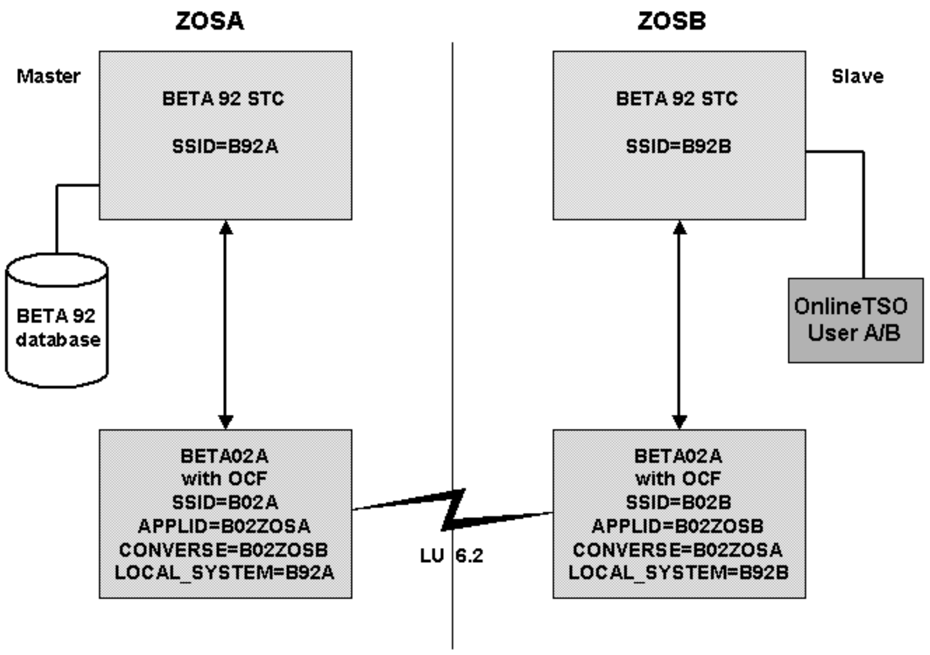
OCF in Beta 02 can also be used to connect an XCF sysplex to an external system (see "Example: Communication between sysplex and non-sysplex" on page 123).

Example 1: X-System Router Beta 02 with slave system

Overview

This example is similar to "Example 1: Host-to-host - 4 STCs and 4 OCF nodes (recommended)" on page 103.

Diagram

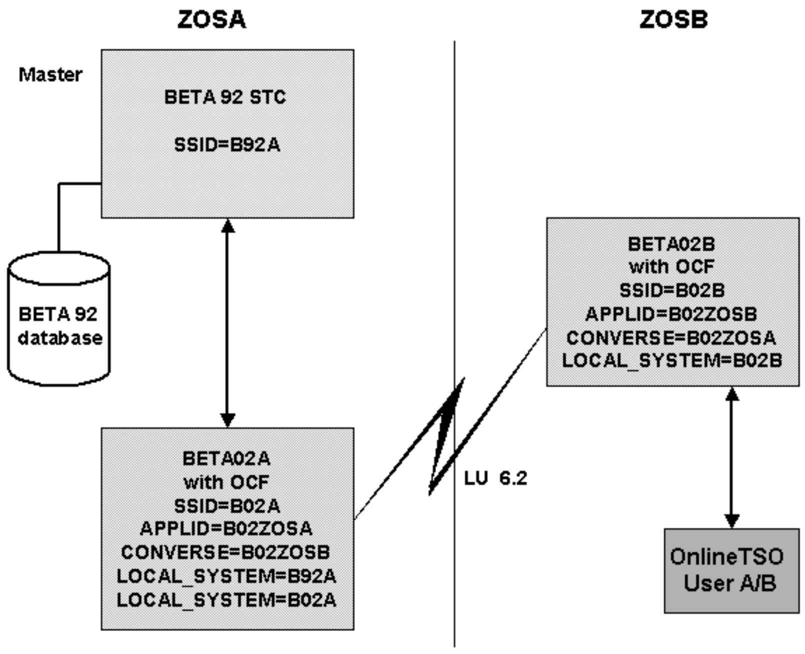


Example 2: X-System Router Beta 02 without a slave system

Overview

The BSA X-System Router enables TSO users and batch jobs on a product started task master system to access another z/OS system without having a product STC as a slave system.

Diagram


ZOSA
ZOSB
Master
BETA 92 STC
SSID=B92A
BETA 92 database
BETA02A with OCF SSID=B02A APPLID=B02ZOSA CONVERSE=B02ZOSB LOCAL_SYSTEM=B92A LOCAL_SYSTEM=B02A
LU 6.2
BETA02B with OCF SSID=B02B APPLID=B02ZOSB CONVERSE=B02ZOSA LOCAL_SYSTEM=B02B
OnlineTSO User A/B

Additional tasks

In this example, two additional tasks need to be completed:

For TSO users:

Adapt the CLIST of the product as follows:

```
ISPEXEC SELECT PGM(BST00PRM)PARM(Bnn,........./DAM)
```

All the connected product master systems are displayed online through parameter /DAM. This parameter must be specified in this form as the last parameter.

For batch jobs:

In order to process batch jobs in the operating system where only the BSA X-System Router Beta 02 is active, the subsystem ID of the product master started task must be available on this operating system, and must be initialized on the LPAR concerned using BST01ARI.

Parameters for OCF via TCP/IP

Overview

Use the X-System-Router Beta 02 to establish OCF communication via TCP/IP. OCF via TCP/IP is controlled via a set of parameters in the LST members that are used by the Beta 02 started task involved.

Remove all LST parameters related to OCF via LU 6.2 from these members, because you can only use either OCF via TCP/IP or OCF via LU 6.2.

Required definitions

In the LST member of each Beta 02 started task that serves as an OCF communication node, code the following LST parameters:

- BSA_OCF_TCPIP_PORT

  This LST parameter activates OCF via TCP/IP and defines the port where the Beta 02 STC receives incoming OCF requests. The OCF port number of each Beta 02 STC must be different because the port number is used as a key.

- BSA_OCF_TCPIP_CONVERSE

  This LST parameter defines the IP address and port number where a CONVERSE partner can be reached. Code BSA_OCF_TCPIP_CONVERSE for each of the other Beta 02 STCs that act as an OCF node within the sysplex.

  BSA_OCF_TCPIP_CONVERSE is required if BSA_OCF_TCPIP_PORT is present. The port defined by BSA_OCF_TCPIP_PORT will not be opened if there are no CONVERSE definitions.

- BSA_OCF_TCPIP_LOCAL_SYSTEM

  This LST parameter defines a local subsystem that you want to be accessible via the local OCF node. Code BSA_OCF_TCPIP_LOCAL_SYSTEM for each local subsystem that is to be accessible to other OCF nodes within the sysplex.

  Unlike OCF LU 6.2, there is no need to define the current Beta 02 subsystem as a local subsystem in its LST member because this information is automatically passed to the partner.

LST parameters

Keyword	Parameter	Description	Option	Default
BSA_OCF_TCPIP_PORT	(*port*,*ipa*, *task*,*rty-intv*, *rty-cnt*)	Use this keyword to activate OCF via TCP/IP; the Beta 02 STC receives all incoming requests via this port and passes each to the ssid included in the request. There are five positional parameters: • *port*, *ipa* and *task* are required • *rty-intv* and *rty-cnt* are optional	optional	none
	port	Port number (max. 5 digits)		
	ipa	IP address (numeric or symbolic, max. 255 chars.)		
	task	Name of the TCP/IP task on the local z/OS system (max. 8 chars.)		
	rty-intv	Retry interval If the opening of the port fails, the STC will try again after the specified interval (default: 60 sec).		
	rty-cnt	Retry counter If the opening of the port fails, the STC will try again for the specified number of times (default: **0** meaning "no limit", i.e. the STC will retry until it succeeds or is stopped).		
BSA_OCF_TCPIP_ CONVERSE	(*port*,*ipa*, *task*,*rty-intv*, *rty-cnt*)	Use this keyword to define one or more partners for OCF via TCP/IP. All send requests are handled via the CONVERSE port(s). The port defined by BSA_OCF_ TCPIP_PORT will not be opened if there are no CONVERSE definitions. There are five positional parameters: • *port*, *ipa* and *task* are required • *rty-intv* and *rty-cnt* are optional	required if BSA_ OCF_ TCPIP_ PORT is coded	none
	port	Port number of the CONVERSE partner (max. 5 digits)		
	ipa	IP address of the CONVERSE partner (numeric or symbolic, max. 255 chars.)		
	task	Name of the TCP/IP task on the local z/OS system (max. 8 chars.)		

Keyword	Parameter	Description	Option	Default
	rty-intv	Retry interval If the opening of the port fails, the STC will try again after the specified interval (default: 5 sec).		
	rty-cnt	Retry counter If the opening of the port fails, the STC will try again for the specified number of times (default: 10).		
BSA_OCF_TCPIP_ LOCAL_SYSTEM	(*ssid*, *productid*)	Use this keyword to define one or more local subsystems that can be reached via the CONVERSE partner. There are two positional parameters; both are required.	optional	none
	ssid	Local subsystem ID		
	productid	Product identifier **BETA*nn*** (for example, **BETA93**)		
BSA_OCF_TCPIP_ CONNECT_TIMEOUT	*time*	Maximum number of seconds that the STC waits for a connection attempt to a CONVERSE partner to succeed. This value applies to all CONVERSE definitions.	optional	60
BSA_TCPIP_TRACE_OCF	YES | NO	Turns the trace function for OCF via TCP/IP on or off. If YES, 8999I messages with additional information are output while a connection is established. This parameter can be updated dynamically via the BSA Service Manager.	optional	NO

Adding an OCF node

Adding a Beta 02 STC as an OCF node requires the following changes:

- In the B02LST*xx* member of the new OCF node:
  - Add an OCF port definition (BSA_OCF_TCPIP_PORT).
  - Add a CONVERSE definition (BSA_OCF_TCPIP_CONVERSE) for each existing OCF node.

  Restart this Beta 02 STC for the changes to take effect.
- In the B02LST*xx* member of each existing OCF node:
  - Add a CONVERSE definition (BSA_OCF_TCPIP_CONVERSE) for the new OCF node.

  Restart these Beta 02 STCs for the changes to take effect.

Removing an OCF node

Removing a Beta 02 STC as an OCF node requires the following changes:

- In the B02LST*xx* member of each remaining OCF node:
    - Remove the CONVERSE definition (BSA_OCF_TCPIP_CONVERSE) of the former OCF node.

  Restart these Beta 02 STCs for the changes to take effect.
- In the B02LST*xx* member of the former OCF node:
    - Remove all OCF-related LST parameters (BSA_OCF_*xxxxxx*).

  Restart this Beta 02 STC for the changes to take effect.

Adding/removing a local subsystem

Adding/removing a local subsystem to/from the OCF communication network requires the following changes:

- In the B02LST*xx* member of the local OCF node:
    - Add or remove the local subsystem definition (BSA_OCF_TCPIP_LOCAL_SYSTEM).

  Restart this Beta 02 started task for the changes to take effect.

Retry mechanism

The retry mechanism of OCF via TCP/IP is analog to the retry mechanism of OCF via LU 6.2.

Retry connect to CONVERSE

If the Beta 02 STC fails to establish a connection to a CONVERSE partner, the Beta 02 STC retries to establish this connection. The maximum number of retries and the time interval between connection attempts are defined via positional parameters in each CONVERSE definition.

The retry mechanism is also triggered when the CONVERSE partner stops a connection. If the Beta 02 STC is unable to establish a connection after the maximum number of retries, it waits for connection attempts from the partner. After the partner has succeeded to establish a connection, the Beta 02 STC reactivates its own CONVERSE connections.

Retry open OCF port

If the Beta 02 STC fails to open the OCF port, the Beta 02 STC retries to open this port. The maximum number of retries and the time interval between connection attempts are defined via positional parameters in the LST parameter BSA_OCF_TCPIP_PORT.

The Beta 02 STC will not retry after the maximum number of retries has been reached. The Beta 02 STC has to be restarted if you want to trigger the restart mechanism.

Retry defaults

The retry mechanism is controlled by several parameters that can be set for each OCF node. We recommend that you **do not change the defaults**.

Displaying OCF connections

You can use the BSA Service Manager or operator commands to display OCF connections. The procedures used for OCF via TCP/IP are the same or similar to the procedures used for OCF via LU 6.2.

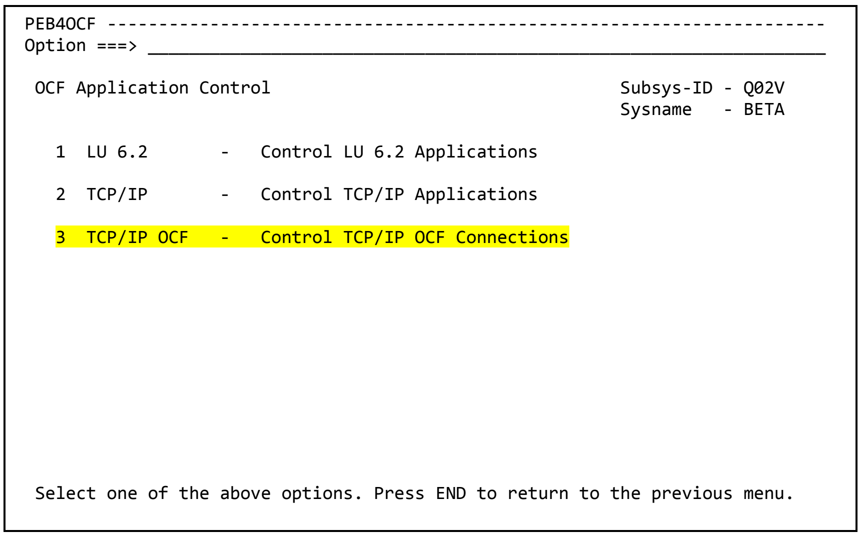
Console command

You can use the following operator console command to display OCF connections:

F *stcname*,D OCF[,ALL]

BSA Service Manager

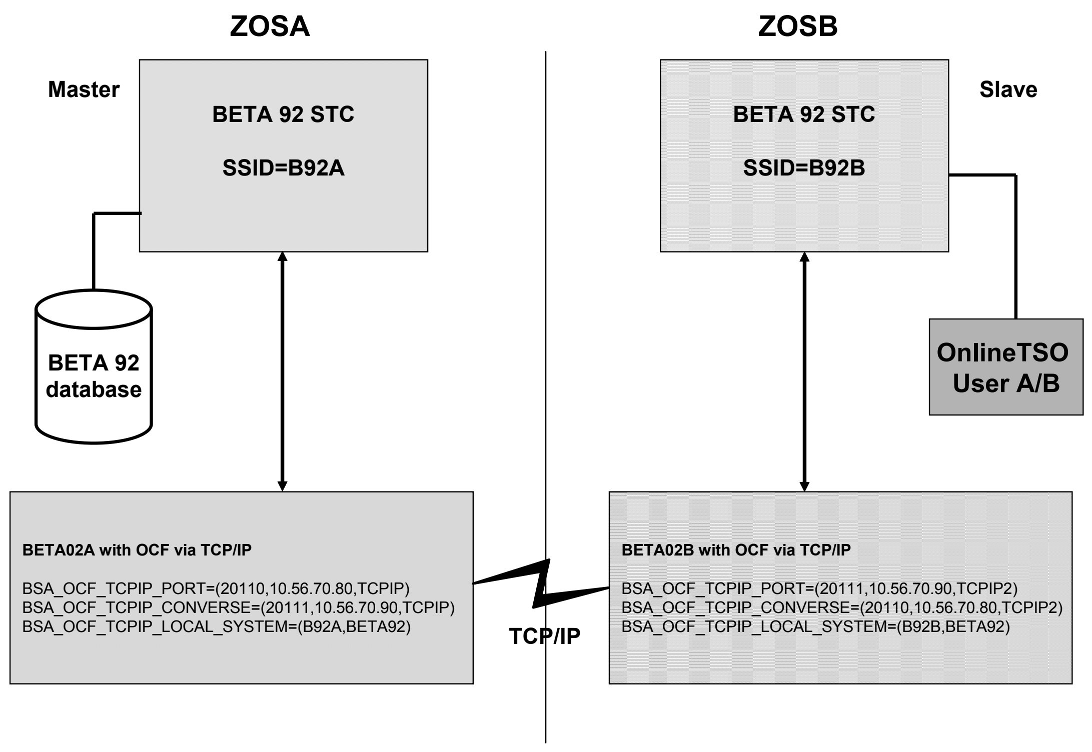
You can use the new BSA Service Manager option **4.2.3** to control and display OCF connections via TCP/IP.



```
PEB4OCF --------------------------------------------------------------------
Option ===> __________________________________________________________________

 OCF Application Control                                   Subsys-ID - Q02V
                                                           Sysname   - BETA

   1  LU 6.2       -    Control LU 6.2 Applications

   2  TCP/IP       -    Control TCP/IP Applications

   3  TCP/IP OCF   -    Control TCP/IP OCF Connections









 Select one of the above options. Press END to return to the previous menu.
```

You can also use the line command **OX** under the BSA Service Manager option **5** to display the OCF via TCP/IP connections of a subsystem.

Example
X-System Router Beta 02
with slave system



```
BSA_OCF_TCPIP_PORT=(20110,10.56.70.80,TCPIP)
BSA_OCF_TCPIP_CONVERSE=(20111,10.56.70.90,TCPIP)
BSA_OCF_TCPIP_LOCAL_SYSTEM=(B92A,BETA92)
```

```
BSA_OCF_TCPIP_PORT=(20111,10.56.70.90,TCPIP2)
BSA_OCF_TCPIP_CONVERSE=(20110,10.56.70.80,TCPIP2)
BSA_OCF_TCPIP_LOCAL_SYSTEM=(B92B,BETA92)
```

Multi-CPU using BSA XCF in a sysplex

In this chapter

Topic	Page
Introduction	117
Which multi-CPU configuration types are possible?	118
Prerequisites and requirements for BSA XCF communication	119
Initializing subsystems in the sysplex	120
Communication with an SSID not in the sysplex	122
Example: Communication between sysplex and non-sysplex	123
Working with the sysplex	124
Displaying XCF members, subsystem and STC info	126
Configuration examples	128

Introduction

Who should read this section?

This section is of interest to systems programmers who want to set up a multi-CPU environment in a sysplex, or who want to provide a communications link between Beta products on different LPARs.

What is BSA XCF?

BSA XCF is the BSA Cross Coupling Facility, which can be used to provide a communications link between Beta products on different LPARs in a sysplex or parallel sysplex. It supplements the current OCF protocol and can even be used to replace it; for example, XCF can be used for communication within a sysplex instead of OCF.

What does BSA XCF do?

BSA XCF communication uses the extended capabilities of the Cross Coupling Facility (XCF) component of the z/OS operating system. It enables Beta products and CPUs to be accessed beyond LPAR borders without requiring the use of an additional task such as the OCF Cross System Router. Beta products and all the connected CPUs are readily available to online users and batch jobs throughout the entire sysplex.

BSA XCF Global Connect

The BSA XCF Global Connect component (see page 131) is a started task with its own address space that remains constantly active in the XCF environment. It facilitates XCF communication for Beta Systems products that do not set up their own Beta Systems product address space in the XCF environment, for example, _beta access.

Which multi-CPU configuration types are possible?

Introduction

BSA XCF works in conjunction with BSA Cross Memory (CM) and OCF.

Configuration types

BSA XCF enables the following configurations:

- Communication within a sysplex using **BSA XCF with BSA Cross Memory (CM)**.
- Communication within the sysplex itself and, in addition, with a system outside the sysplex. This configuration uses **BSA XCF with BSA Cross Memory (CM) in conjunction with OCF**.

BSA XCF with BSA Cross Memory (CM)

In the **BSA XCF with BSA Cross Memory (CM)** configuration, Beta subsystems on different LPARS communicate with one another via the BSA XCF protocol. Beta subsystems and users located on the same LPAR are connected by BSA CM, i.e. via the Beta SVC. XCF also makes it possible to interconnect different Beta products (e.g. B93, B97).

BSA XCF in conjunction with OCF

In **BSA XCF in conjunction with OCF**, Beta subsystems connected within a sysplex by means of BSA XCF/CM can also be given access to an LPAR that is not part of the sysplex. The connection between the sysplex and the external LPAR is set up via OCF.

SSID search order

BSA searches for a requested subsystem ID in the following sequence:

1. CM (i.e. on the local LPAR where the user or batch job is currently working)
2. XCF (i.e. within the BSA XCF complex)
3. OCF (i.e. on the active OCF complex)

Prerequisites and requirements for BSA XCF communication

Overview

Please ensure that the prerequisites for BSA XCF are met under all circumstances. Failure to do so may produce unexpected results for online users, or cause requests in batch jobs to hang for no apparent reason.

BSA level and SVC level

BSA V4 or higher must be installed.

Operating system

All the LPARs (systems) involved must be logically connected via IBM XCF, i.e. they must be defined in a sysplex or parallel sysplex.

See the IBM manual *Setting Up a Sysplex* for the installation/activation of these connections.

Length of SSID

Subsystem IDs must always have a length of exactly four characters. SSIDs of any other length cannot be used in a BSA XCF sysplex.

Unique SSID?

There are different requirements for subsystem IDs, depending on the type of configuration. Please remember that the requirements for different Beta products may also vary.

Configuration	Same SSID names	Unique SSID names
Shared DB	Possible*	Possible
Master/Slave	Not allowed	Required
Several masters	Not allowed	Required

* The last subsystem to be started after an IPL is always selected when shared databases with the same SSID names are used. This means that all requests go through this subsystem.

XCF optimization

To achieve optimum signaling performance, XCF group BETAGRP must be assigned to an XCF transport class and the maximum length of the data in the class must be defined. The maximum length of BSA messages is 35288 bytes. The keyword parameters can be set at the time of IPL when the CLASSDEF, MAXMSG, PATHIN, PATHOUT, LOCALMSG statements in the COUPLExx parmlib members are defined, or by dynamic allocation in the operator command SETXCF. A detailed description of the required parameters and the dynamic operator command can be found in the IBM manuals *Setting up a Sysplex* and *Operator Commands* respectively.

Note: When defining CLASSLEN, please remember that XCF adds other information to the actual BSA message, depending on your configuration. You will find detailed information on tuning the message buffer size and class length in the IBM manual *Setting up a Sysplex*.

Initializing subsystems in the sysplex

Overview

A BSA XCF capable subsystem must be defined and initialized on each LPAR where this subsystem is to be available.

Initializing subsystems

BSA XCF capable subsystems are initialized in the usual way with BST01ARI (see "BST01ARI: Initializing subsystem and security environment" on page 54).

In addition to the usual parameters for the program, the parameters XCF=YES and XCF_PROD=*nn* must be defined in member B*nn*SSI*xx* for BSA XCF initialization.

The following message is issued when initialization has been successfully completed:

```
xxx9170I XCF - SUPPORT HAS BEEN ACTIVATED
```

Other parameters

The parameter GOTO_OCF=*ssid* only needs to be defined when initializing the non-sysplex subsystem on the LPARs in the sysplex. *ssid* is the SSID of the system inside the sysplex that contains the OCF parameters for communication with the system that is outside of the sysplex (see "Communication with an SSID not in the sysplex" on page 122).

For information on when to use the XCF_GBAS parameter, see "Introducing BSA XCF Global Connect" on page 131.

B*nn*SSI*xx* parameters for XCF initialization

Parameter name	Value	Description	Default
XCF	YES | NO	YES Activates BSA XCF support NO Deactivates BSA XCF support	NO
XCF_PROD	*nn*	Two-digit product identifier of the subsystem This parameter is required if XCF = YES.	none
XCF_GBAS		Enables the automatic start of the BSA XCF Global Connect component (see page 131). There are two possibilities:	none
	name	where *name* is the name of the started task that provides the address space. You can find a sample procedure in the BSA.SAMPLIB in member BETAGXCF.	
	-OR-		
	IEESYSAS.*name*	where *name* is an identifier added to IEESYSAS to enable the started task to be identified and controlled.	
GOTO_OCF	*ssid*	This parameter is only required if you want to access a subsystem that is outside of the sysplex. Here, ssid is the SSID of the subsystem inside the sysplex that contains the OCF parameters enabling users (=TSO users, batch jobs, subsystems) in the sysplex to access the non-sysplex subsystem. Do **not** set this parameter when initializing any of the subsystems that are to be active **inside** the sysplex.	none

Deinitialization

To redefine a BSA XCF capable subsystem ID as NOT BSA XCF capable:

1. Specify XCF=NO in the active B*nn*SSI*xx* member (or remove or comment out this parameter).
2. Run BST01ARI on all affected LPARs of the sysplex to reinitialize this subsystem.

Communication with an SSID not in the sysplex

Overview

When you are working in a sysplex, you can also access and work with a subsystem (SSID) that is not part of that sysplex. In this case, communication takes place via OCF between one SSID inside the sysplex and an SSID outside of the sysplex.

OCF parameters

OCF parameters must be set in the usual way in the LST members of each of the two subsystems: the subsystem inside the sysplex that is to be used to communicate with the outside, and the subsystem outside the sysplex. The OCF parameters can be set in the external subsystem itself, or in the BSA X-System Router Beta 02. We recommend that you use Beta 02. For more information, see "Establishing OCF communication via X-System Router (Beta 02)" on page 107.

Defining OCF_LOCAL

All the sysplex subsystems that are to be reached by the external subsystem must be defined as OCF_LOCAL on the sysplex-internal subsystem that is to be used to communicate with the outside. All the LPARS in the sysplex are then treated as a single LPAR by the OCF router task.

Initialization in the sysplex

The subsystems inside the sysplex must be defined on each of the LPARs in the sysplex and initialized with the following parameters:

```
XCF=YES
XCF_PROD=product
```

The SSID of the external subsystem (i.e. the one outside the sysplex) must be defined on each of the LPARs in the sysplex and initialized with the following parameters:

```
XCF=YES
XCF_PROD=product
GOTO_OCF=ssid
```

This enables access to the external non-sysplex subsystem from the subsystems within the sysplex.

Do NOT define parameter GOTO_OCF when initializing the subsystems that are defined as part of the sysplex.

Example

"Example: Communication between sysplex and non-sysplex" on page 123 clearly illustrates the settings that need to be made and where they need to be made.

Example: Communication between sysplex and non-sysplex

Overview

The following table shows an example of the settings that need to be made inside and outside the sysplex to enable communication between subsystem B*nn*C (SSID inside the sysplex) and subsystem B*nn*D (SSID outside the sysplex).

Note: Parameter GOTO_OCF is only set when initializing the **non**-sysplex subsystem on the LPARs in the sysplex. Do **not** set this parameter when initializing any of the subsystems that are active within the sysplex (in this example SSIDs B*nn*A, B*nn*B and B*nn*C).

Inside the sysplex	**Outside the sysplex**

LPAR1 => SSID = B*nn*A
LPAR2 => SSID = B*nn*B
LPAR3 => SSID = B*nn*C
	LPAR4 => SSID = B*nn*D

All of the subsystems B*nn*A, B*nn*B and B*nn*C are defined and initialized inside the sysplex on each of the LPARs 1, 2 and 3 (see "Initializing subsystems in the sysplex" on page 120).
Parameters for BST01ARI in B*nn*SSI*xx*: As described by the product documentation, with the following XCF parameters:
`XCF=YES`
`XCF_PROD=nn`
	Subsystem B*nn*D is defined and initialized outside the sysplex on LPAR 4. Parameters for BST01ARI in B*nn*SSI*xx*: As described by the product documentation, **without** XCF parameters.

Subsystem B*nn*D is defined and initialized inside the sysplex on LPARs 1, 2 and 3, using the following parameters:
`XCF=YES`
`XCF_PROD=nn`
`GOTO_OCF=BnnC`
(where B*nn*C is the SSID of the subsystem in the sysplex that contains the OCF parameters that enable communication with the subsystem outside of the sysplex).	

The following parameters are set in the LST member of B*nn*C:
`OCF_APPLID=APPLnnC`
`OCF_CONVERSE=APPLnnD`
`OCF_LOCAL_SYSTEM=BnnC,BETAnn`
`OCF_LOCAL_SYSTEM=BnnA,BETAnn`
`OCF_LOCAL_SYSTEM=BnnB,BETAnn`
	The following parameters are set in the LST member of B*nn*D:
	`OCF_APPLID=APPLnnD`
	`OCF_CONVERSE=APPLnnC`
	`OCF_LOCAL_SYSTEM=BnnD,BETAnn`

Working with the sysplex

Before you start the subsystem

Before you can start BSA XCF capable subsystem, you have to initialize it (see "Initializing subsystems in the sysplex" on page 120).

Joining the sysplex

When a subsystem that is BSA XCF capable is activated, message 9170I appears, and then it is joined to XCF. As a rule, the join is made in the XCF group BETAGRP. When the join is executed, a unique member name is established for each joined user (started task, batch job or TSO online user), identifying this user to XCF.

Note: The XCF group must have an assignment to an XCF transport class. For further information, see "XCF optimization" on page 119.

XCF member name

The member names created when a join is set up to XCF group BETAGRP have the following structure:

```
Bnn$sctssidasidn
```

where:

Character...	Meaning...
nn	Beta product number that was specified by parameter XCF_PROD when the subsystem ID was initialized
$	Separator
sc	Sysclone name of the system from which the join was executed
t	Type of address space: S Started task J Batch job T TSO online user R Started task that works like a batch job with this ssid
ssid	Subsystem ID the join was executed for
asidn	ASID number (decimal) of the address space

Searching for subsystem IDs

When a request is made to a subsystem ID, XCF searches for it in the following order of sequence (the subsystem concerned must be active):

1. The search for the subsystem ID is first made on the local LPAR (i.e. the LPAR the user or the batch job is currently working on).
2. If the subsystem ID cannot be found on the local LPAR, the search is continued within the BSA XCF complex.
3. If the subsystem ID cannot be found in the BSA XCF complex, the next and final search is made on the active OCF complex. If this search also fails, the request made to the subsystem ID terminates with an error.

Displaying XCF members, subsystem and STC info

Displaying members

To display all the members connected to XCF via the XCF group BETAGRP, use the following operator command:

```
D XCF,GRP,BETAGRP,ALL
```

The following is written to the operator console and the SYSLOG:

```
+----------------------------------------------------------------------+
|IXC333I  11.33.50  DISPLAY XCF 638                                    |
|      INFORMATION FOR GROUP BETAGRP                                   |
|      MEMBER NAME:        SYSTEM:      JOB ID:      STATUS:           |
|      B92$50SS92P00302     BT50         BETA92P      ACTIVE            |
|      B92$50TS92P00056     BT50         USER2        ACTIVE            |
|      B92$50TS92P00304     BT50         USER1        ACTIVE            |
|      B92$90TS92P00046     BT90         USER1        ACTIVE            |
+----------------------------------------------------------------------+
```

Displaying available subsystems

There are two ways of displaying all the available subsystems of a product:

- Using the product option **P.2**
- Using the BSA Service Manager option **D.S.5**

The display shows the status of the subsystem (Local, OCF or XCF, L O X D in the table), and the name of the system where the subsystem ID is active.

Display using P.2

The following panel is displayed under option **P.2**:

```
PEB0SYST --------------------------------------------------------- Row 1 of 14
Command ===> ________________________________________________ Scroll ===> PAGE

 System Selection Table

  S - Select System

Sel Name     Location         SSID Sysname  Product Version PTF Lvl Act L O X D
    Title
    PROD     BERLIN           B92P BETA     BETA92  VxRx-00 POMnnnn YES Y N N
    BETA92 PRODUCTION SYSTEM
-------------------------------------------------------------------------------
    TRAINING BERLIN           D92P BETA     BETA92  VxRx-04 POMnnnn YES Y N N
    BETA 92 TRAINING SYSTEM
-------------------------------------------------------------------------------
    B92VxxM  BERLIN           92T4 BETA     BETA92  VxRx-06 POMnnnn YES Y N N
    B92 Vxx0 DEVELOP SYSTEM 1
-------------------------------------------------------------------------------
    B92VxxM  BERLIN           92T2 BETA     BETA92  VxRx-00 POMnnnn YES Y N N

-------------------------------------------------------------------------------
    BSA      BERLIN           S92P BETA     BETA92  VxRx-00 POMnnnn YES Y Y N S
    BETA92 PRODUCTION SYSTEM FOR BSA
-------------------------------------------------------------------------------
```

Display using D.S.5

The following panel is displayed under option **D.S.5**:

```
PEB4SID ------------------------------------------------------- Row   1 of  98
Command ===> ________________________________________________ Scroll ===> PAGE

 Display Beta-STC Information Overview                 Subsys-ID - B92P
                                                       Sysname   - BETA

  OC - Control LU 6.2 Applications OX - OCF/XCF Connections
  TC - Control TCP/IP Applications XT - Display XCF-STC Connection
  XM - XCF Memberinformation       SE - Security Exits
  RB - RSB Information             SI - Beta Subsystem Information
  S  - Select a Subsystem

Sel   SSID A Jobname  Jobnumber System   PI Location         Name     L O X D
      BQLA Y BQLN2    S0042025  BETA     05                           Y N Y M
      BSAP Y BETA04AP S0034187  BETA     17                           Y N N M
      B02P Y BETA02P  S0000065  BETA     02                           Y N N S
      B02Z Y B92T2B02 S0000118  BETA     02                           Y N N S
      B07P Y BETA07P  S0000068  BETA     07                           Y N N S
      B09P Y BETA09P  S0001436  BETA     09                           Y N N M
      B48A Y BETA48A  S0006222  BETA     48 BERLIN           B48V450  Y N N M
      B48D Y BETA48D  S0000121  BETA     48 BERLIN           B48V440  Y N N M
      B77P Y BETA77P  S0000015  BETA     77 BERLIN           PROD     Y N N M
      B88H Y BETA88H  S0049534  BETA     88 BERLIN           TESTWW   Y Y N M
      B88K Y BETA88K  S0008527  BT40     88 BERLIN           BT40     N Y N S
```

Subsystem unavailable

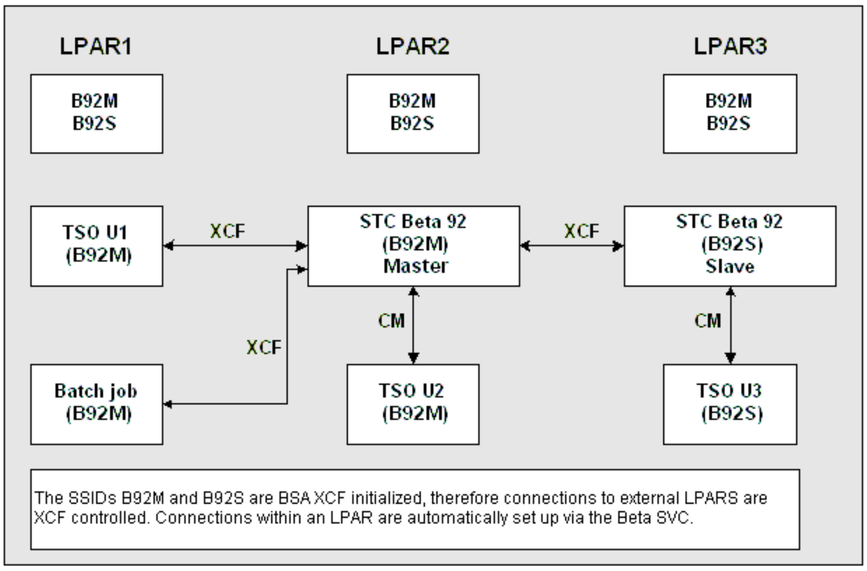
The message "Subsystem unavailable" appears if the selected subsystem is not active or if it cannot be reached. The message includes a return code (RC) and a reason code (REASON). Press PF1 to find out the exact reason or cause. For information on these return/reason codes, see "Subsystem connection errors" in *BSA Messages and Codes*.

```
PEB0PRF ----------------------------------------------- Subsystem unavailable
Command ===> _________________________________________________________________
MEBSF521 - Subsystem is not active - RC: 00000004  REASON: 20810006
 BETA System Profile Options

  System Name          ===>
  System Location      ===>
  Subsystem ID            : ssid
  System Level            : VxRx-nn      BSA Level      : nnnn-nn
  System PTF Level        : xxxnnnn      BSA PTF Level  : PBSnnnn


  User Date Mask       ===> MM/DD/YY     MM/DD/YY, DD.MM.YY, DD/MM/YY, YY.DDD
                                         MM/DD/YYYY, DD.MM.YYYY, DD/MM/YYYY
                                         YYYY.DDD, YYYY-MM-DD
  BETA Product Language ===> E           (E)nglish,(G)erman

  Extended Help Mode   ===> YES          (Y)es, (N)o



 Press the ENTER key to update your system profile options.
 Press the END key to return to the previous menu.
```

Configuration examples

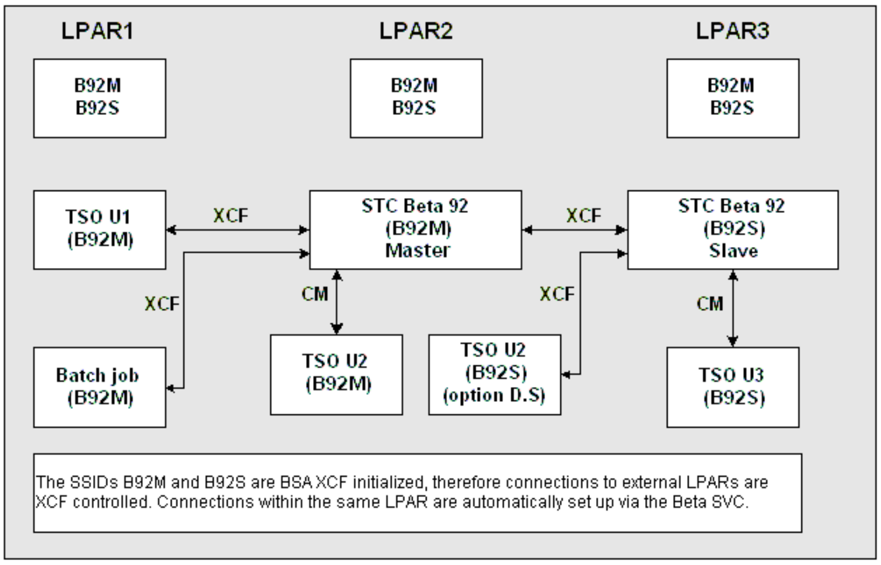
CM = Cross Memory Communication via SVC.

Master-slave connection in a sysplex (1)



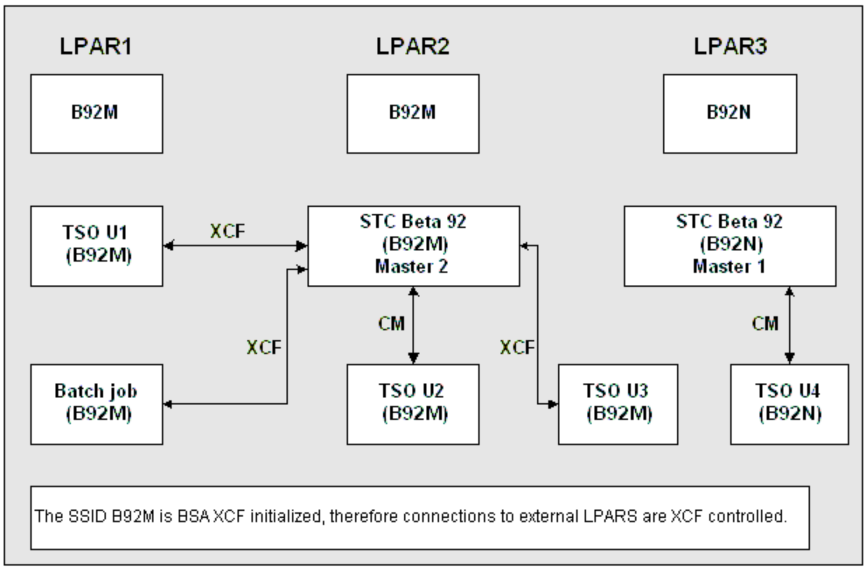
The SSIDs B92M and B92S are BSA XCF initialized, therefore connections to external LPARS are XCF controlled. Connections within an LPAR are automatically set up via the Beta SVC.

Master-slave connection in a sysplex (2)



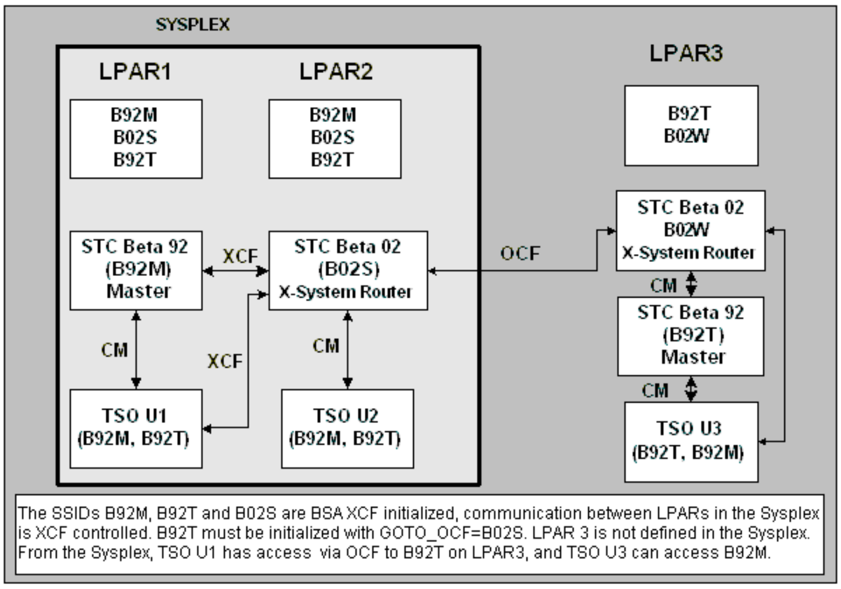
The SSIDs B92M and B92S are BSA XCF initialized, therefore connections to external LPARs are XCF controlled. Connections within the same LPAR are automatically set up via the Beta SVC.

Shared databases with different master SSIDs



The SSID B92M is BSA XCF initialized, therefore connections to external LPARS are XCF controlled.

Communication outside the sysplex



The SSIDs B92M, B92T and B02S are BSA XCF initialized, communication between LPARs in the Sysplex is XCF controlled. B92T must be initialized with GOTO_OCF=B02S. LPAR 3 is not defined in the Sysplex. From the Sysplex, TSO U1 has access via OCF to B92T on LPAR3, and TSO U3 can access B92M.

Creating an address space with BSA XCF Global Connect

In this chapter

Topic	Page
Introducing BSA XCF Global Connect	131
System overview of BSA Global XCF	133
Activating BSA XCF Global Connect	134

Introducing BSA XCF Global Connect

Overview

The BSA XCF Global Connect component is a started task with its own address space that remains constantly active in the XCF environment.

Beta Systems product components that do not set up their own address space in the XCF environment can use the address space of BSA XCF Global Connect, which has the following advantages:

- XCF communication is facilitated.
- Load on resources is reduced.

When to use

Use BSA XCF Global Connect component when the Beta SMF exit communicates via BSA XCF.

The Beta SMF exit is used by _beta access and _beta access monitor.

Note: The following warning message will occur if the Global XCF Connect address space is required, but not active:

*xxx*9116W GLOBAL XCF CONNECT FAILED - *nnnnnnnn*

Background

BSA XCF enables Beta products to access the various LPARs in a sysplex without having an additional started task on the LPAR concerned. When an address space sends a request to the other LPAR, a connection is set up to XCF in the operating system. The requestor is usually a TSO user, a batch job, or a slave started task. As a rule, all address spaces are Beta address spaces and are controlled by BSA XCF communication. When the address space (member) leaves the group, the connection is terminated properly by means of IXCLEAVE.

Some Beta Systems applications use components which, although they can communicate via BSA XCF, do not run in their own address spaces. The SMF exit used by _beta access and _beta access monitor is an example of such a component. This means that although a JOIN can be made (in any address space in z/OS), it is not possible to terminate the JOIN by means of a Beta-controlled IXCLEAVE. Each activation of an address space could in this case lead to the writing of a LOGREC record. Another problem could be an increased amount of overhead and the loss of performance because each address space receives its own XCF member in this type of communication.

Using the additional Beta-controlled address space provided by the BSA XCF Global Connect component overcomes both problems.

Features of BSA XCF Global Connect

Generation of a separate address space for BSA XCF communication:

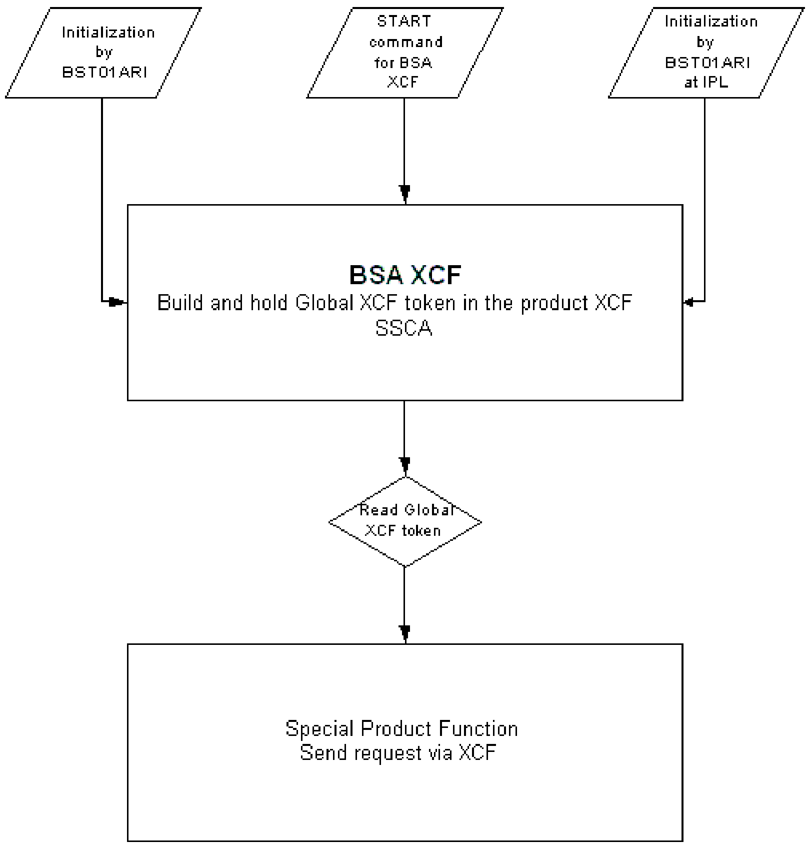
- Always active and cannot be canceled
- Easy to install and activate
- Is used automatically when the product sends a request from the SMF exit
- Can be controlled by various operator commands and product functions, for example, from _beta access via the SMF exit

Limitations

The permanently active address space can only be used for special requests from the SMF exit. A valid Global XCF token must always be available. If the XCF token for BSA XCF Global Connect is not valid when a request is sent, no attempt will be made to generate a new XCF sender token, unlike basic BSA XCF communication. The request will always be refused and no reJOIN will be made to XCF.

System overview of BSA Global XCF

System overview


Initialization by BST01ARI
START command for BSA XCF
Initialization by BST01ARI at IPL
BSA XCF
Build and hold Global XCF token in the product XCF SSCA
Read Global XCF token
Special Product Function
Send request via XCF

Activating BSA XCF Global Connect

Overview

The BSA XCF Global Connect component is a separate address space (started task or batch job) that must remain constantly active to enable communication. This address space sets up an IXCJOIN to XCF and then administers the XCF token it receives. The XCF token is used for communication.

Activation

The name of the sample XCF Global Connect started task is BETAGXCF. The address space can only be activated once, any subsequent attempts will be rejected. There are two ways of activating BSA XCF Global Connect:

- By means of an operator start command
- During the initialization of a subsystem via BST01ARI

Method 1: Operator start command

BSA XCF Global Connect can be activated by an operator start command for the started task (for example, BETAGXCF), either when the operating system is already active, or at the time of IPL by means of member COMMND*xx* in the SYS1.PARMLIB. The necessary procedure must be available.

Method 2: BST01ARI

BSA XCF Global Connect can be activated during the initialization of a subsystem via the program BST01ARI, either when the operating system is already active, or at the time of IPL:

1. Enter the subsystem ID of the subsystem to be initialized in member IEFSSN*xx*.
2. Specify the XCF_GBAS parameter in the B*nn*SSI*xx* member of the subsystem that is to be initialized.

The BSA XCF Global Connect address space is started if it is not already active. Activation of the address space is complete when the XCF token has been generated and saved. If the address space is already active, the parameter has no negative effect. The address space is not restarted.

The XCF token is determined on the basis of the name/token pair generated when the address space is started. If this is not possible, the token is taken from the first subsystem found that has already been initialized for the BSA XCF Global Connect component.

Initialization parameter

Code the XCF_GBAS parameter in the B*nn*SSI*xx* LST member if BST01ARI is to activate the BSA XCF Global Connect component during subsystem initialization.

Parameter	Req/Opt	Description
XCF_GBAS	optional	Name of a started task. There are two possibilities: • You can specify the name (max. 8 characters) of the started task procedure that provides the address space. The procedure must be available in the appropriate procedure library. You can find a sample procedure in the BSA.SAMPLIB in member BETAGXCF. • You can specify IEESYSAS as the started task procedure that provides the address space. It must be specified as follows: `IEESYSAS.`*`name`* where *name* is a name given to the address space to enable commands to be executed on it. In this case no separate procedure is required.

Address space control

The following operator commands are available:

Command	Description
`STOP` *`stcname`*	Stops the address space. A message requests confirmation of the command.
`MODIFY` *`stcname`*`,STOP`	Stops the address space. A message requests confirmation of the command.
`MODIFY` *`stcname`*`,REFRESH`	Deletes and then rebuilds all the relevant information concerning the address space (new XCF token, re-initialization of the subsystem with the new token). A message requests confirmation of the command.
`MODIFY` *`stcname`*`,DISPLAY`	Displays the XCF token used and the XCF member.
`MODIFY` *`stcname`*`,SHOW`	Displays all the subsystems that have been initialized with the XCF token.

BSA TCP/IP server

In this chapter

Topic	Page
Introduction	137
Server functions and features	138
BSA TCP/IP server ports	139
Compression and encryption for the BSA TCP/IP server	141
BSA TCP/IP server logon exit (B02UXSIN)	142
BSA TCP/IP server as a standalone STC (Beta 02)	144
BSA TCP/IP server in a Beta product STC	146
BSA TCP/IP server as an RFF job for EDF	147
SFF client groups for a TCP/IP port	148
Operator commands for the TCP/IP server	150
LST parameters for the BSA TCP/IP server	154
Securing TCP/IP connections with AT-TLS	163
Configuring Policy Agent as a started task in z/OS	164
Defining the security authorization for Policy Agent	165
Defining the Policy Agent configuration files	166
Configuring AT-TLS	167
Defining the AT-TLS policy rules	169
Creating and configuring digital certificates in RACF	172
Securing a connection between BSA TCP/IP server and client partner	173

Introduction

Overview

The BSA TCP/IP Server enables cross-platform communication between different systems, for instance, between UNIX and z/OS, and allows online access from Web applications, for example. It is a concurrent server. This means that it can serve more than one client at a time by performing tasks in interleaved intervals. It can be used in common by all Beta products and their add-ons, for example, Web Enabler or EDF. The BSA TCP/IP server consists of a request router, the concurrent BSA TCP/IP server itself and its own server client.

Environment

The BSA TCP/IP server runs in an SFF (Subsystem Function Facility) environment and – under certain conditions – also in an RFF (Remote Function Facility) environment.

Server compatibility

The server is compatible with all Beta products and add-ons, i. e. the server can be activated dynamically without having to change anything within the Beta product or add-on itself.

Port types

The BSA TCP/IP server supports global ports and application ports.

Server functions and features

Dynamic server control

Various z/OS MODIFY commands are available, including

- Start and stop functions for the server
- Refresh server
- Display server status
- Display status of users logged onto a Beta product via the server

Ports

Several TCP/IP servers with different ports can be activated using the request router, no matter which Beta product owns the port. Different types of port are available: Global (GLB), Application (APPL) and EDF. Identical ports cannot have different IP addresses.

- GLB: A global port enables different add-ons of one or more products to run over the same port.
- APPL: An application port enables one application of one product to run over its own port.

Note: The add-on Extended Input requires an application port. Extended Input cannot communicate via a global port.

IPv6, IPv4, and DNS support

The BSA TCP/IP server supports both IPv6 addresses and IPv4 addresses.

It includes domain name server (DNS) support for the resolution of symbolic IP addresses (canonical notation) to numeric IP addresses. The maximum length of a symbolic IP address is 255 characters.

You can control how symbolic IP addresses are to be resolved (IPv6 or IPv4 or both) and which reachable addresses are to be used (first or all).

Administration

Administration of users logged onto the system via the server, and monitoring the server when time limits have been exceeded.

Automatic retry function

Automatic retry is activated when, for example, the requested TCP/IP started task is not active in z/OS, or when the connection to the task has been interrupted due to an error.

Exits

A central security exit for user logon can be activated.

Encryption/compression

Encryption and/or compression are supported for data transfer via TCP/IP.

Message routing

Automatic message routing to a Beta product when the BSA TCP/IP server has been activated as a standalone started task (Beta 02). This function depends on the functionality of the Beta product concerned.

BSA TCP/IP server ports

Port types

The BSA TCP/IP server supports these types of ports:

- **Global port**

  A global port is available to the add-ons of one or more Beta products. A global port of the BSA TCP/IP server can be shared between products and add-ons.

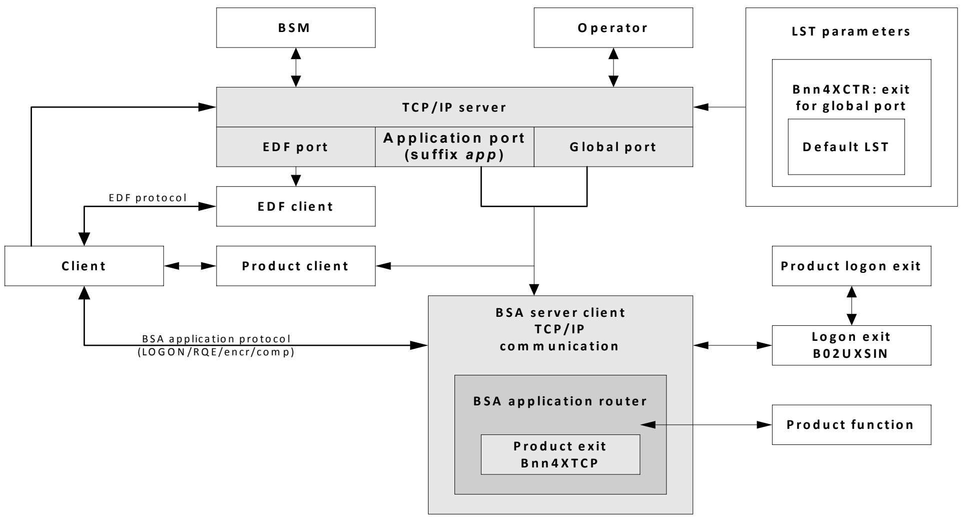
- **Application port**

  An application port is available to the specified application (add-on) of one product. An application port cannot be shared between products and add-ons.

Both global and application ports can be used in the same configuration.

Important: Extended Input (EDF) requires an application port. This add-on cannot communicate via a global port.

Communication overview



Port definitions

An **application port** is defined via the keyword B*nn*_TCPIP_PORT_*app*, where *nn* is the number of the product and *app* is the 3-digit abbreviation of the add-on, for example, B91_TCPIP_PORT_OSY, B92_TCPIP_PORT_OSY, or B93_TCPIP_PORT_BWE.

Use an application port if this port should be used exclusively by this application, for example, because you want to control this application differently from other applications.

A **global port** is defined via the keyword B*nn*_TCPIP_PORT (without application suffix).

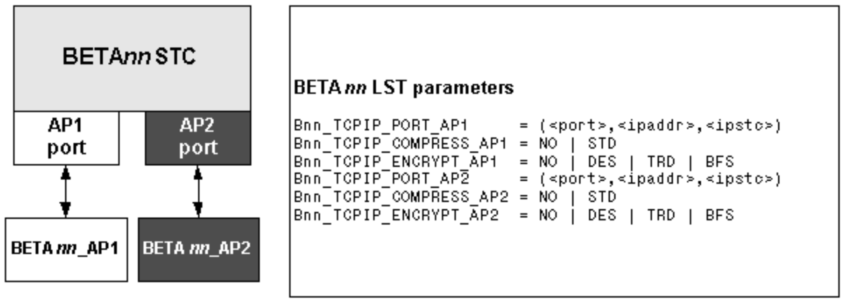
Encryption and compression

Encryption can be controlled via the LST parameters B*nn*_TCPIP_ENCRYPT and B*nn*_TCPIP_ENCRYPT_*app*. Compression can be controlled via the LST parameters B*nn*_TCPIP_COMPRESS and B*nn*_TCPIP_COMPRESS_*app*.

Use B*nn*_TCPIP_ENCRYPT and B*nn*_TCPIP_COMPRESS if you want to set values for all applications of one product. Use B*nn*_TCPIP_ENCRYPT_*app* and B*nn*_TCPIP_COMPRESS_*app* if you want to set values for an individual application of one product.

Example 1: Application ports

Following is an example of how the LST member of Beta *nn* could be configured to provide application ports for two applications (add-ons), AP1 and AP2:

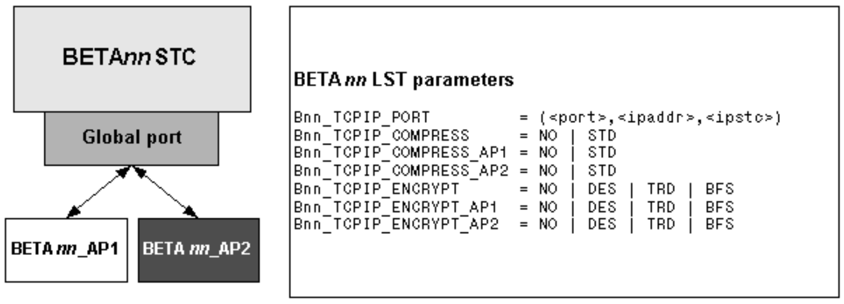


BETA *nn* LST parameters

```
Bnn_TCPIP_PORT_AP1     = (<port>,<ipaddr>,<ipstc>)
Bnn_TCPIP_COMPRESS_AP1 = NO | STD
Bnn_TCPIP_ENCRYPT_AP1  = NO | DES | TRD | BFS
Bnn_TCPIP_PORT_AP2     = (<port>,<ipaddr>,<ipstc>)
Bnn_TCPIP_COMPRESS_AP2 = NO | STD
Bnn_TCPIP_ENCRYPT_AP2  = NO | DES | TRD | BFS
```

Example 2: Global port

Following is an example of how the LST member of Beta *nn* could be configured to provide a global port for two applications (add-ons), AP1 and AP2:

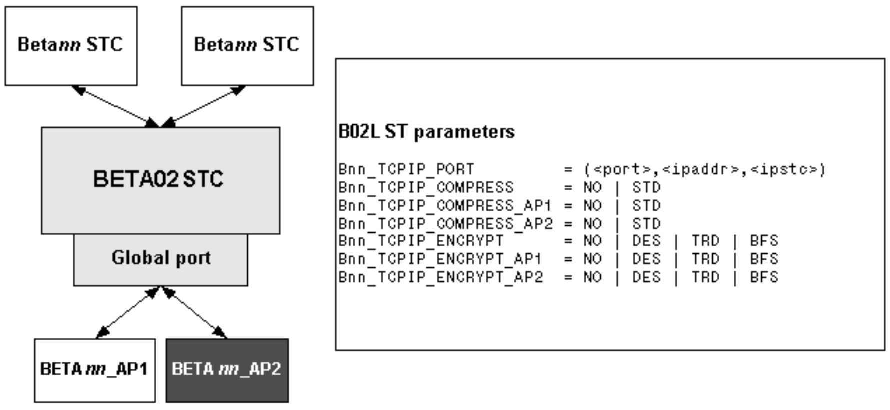


BETA *nn* LST parameters

```
Bnn_TCPIP_PORT         = (<port>,<ipaddr>,<ipstc>)
Bnn_TCPIP_COMPRESS     = NO | STD
Bnn_TCPIP_COMPRESS_AP1 = NO | STD
Bnn_TCPIP_COMPRESS_AP2 = NO | STD
Bnn_TCPIP_ENCRYPT      = NO | DES | TRD | BFS
Bnn_TCPIP_ENCRYPT_AP1  = NO | DES | TRD | BFS
Bnn_TCPIP_ENCRYPT_AP2  = NO | DES | TRD | BFS
```

Example 3: Global port in Beta 02

Following is an example of using a global port in Beta 02:



B02L ST parameters

```
Bnn_TCPIP_PORT         = (<port>,<ipaddr>,<ipstc>)
Bnn_TCPIP_COMPRESS     = NO | STD
Bnn_TCPIP_COMPRESS_AP1 = NO | STD
Bnn_TCPIP_COMPRESS_AP2 = NO | STD
Bnn_TCPIP_ENCRYPT      = NO | DES | TRD | BFS
Bnn_TCPIP_ENCRYPT_AP1  = NO | DES | TRD | BFS
Bnn_TCPIP_ENCRYPT_AP2  = NO | DES | TRD | BFS
```

Compression and encryption for the BSA TCP/IP server

Overview

The BSA compression and encryption feature for TCP/IP is available on z/OS and UNIX platforms. Designed exclusively for data exchange, it is based on the internal BSA communication interface, i.e. on special BSA control information that is put in front of the actual data. BSA encryption and compression are only available on ports without SSL.

Data exchange can take place:

- Between UNIX/NT platforms and z/OS
- Between the Web Enabler application server and z/OS

How it works

Compression and encryption are performed in:

- BSA TCP/IP server on z/OS
- BQL server on all BSA-supported UNIX platforms

During the procedure, the client and the corresponding server decide which protocol is to be used during the TCP/IP connect of the client. Thus the client adapts itself automatically to the supported platform. As a rule, the server decides how data are to be transmitted.

ZIF (zIIP) support

Under z/OS, compression and decompression can be offloaded to IBM's z9 Integrated Information Processor, thus optimizing the use of available resources. For more information, see "zIIP Facility (ZIF)" on page 211.

Controlling compression/ encryption

Compression and encryption can be controlled via the following LST parameters, which must be specified in the system where the TCP/IP server is running:

- B*nn*_TCPIP_COMPRESS[_*app*] = STD | NO | YES

  Specify STD or YES to use Beta standard compression.
  NO means no compression, which is the default.

- B*nn*_TCPIP_ENCRYPT[_*app*] = DES | TRD | AES | BFS | NO

  Specify the encryption algorithm:
  DES for the DES algorithm
  TRD for the triple DES algorithm
  AES for advanced encryption standard
  BFS for the Blowfish algorithm
  NO means no encryption, which is the default.

Code B*nn*_TCPIP_ENCRYPT and B*nn*_TCPIP_COMPRESS if you want to set values for all applications of one product. Code B*nn*_TCPIP_ ENCRYPT_*app* and B*nn*_TCPIP_COMPRESS_*app* if you want to set values for an individual application of one product.

BSA TCP/IP server logon exit (B02UXSIN)

Overview

The logon exit of the BSA TCP/IP server must be activated to enable a check to be made on whether a specific user is allowed to log onto the z/OS system.

The exit must be used for all products except Beta 09 (VDF).

Activation

The exit is provided in source form in member B02UXSIN in the BSA.SAMPLIB.

To activate the exit:

1. Assemble and link the exit into the BETA.APFLOAD used by the product.

   A sample compile job can be found in member BSTUXASM in the BSA.SAMPLIB. You can also use the tailored job in member G#02XSIN in the BSA.CNTL.
2. Initialize the product with the exit (B*nn*INIT with BST01ARI).

Note: We recommend that you do **not** change the name B02UXSIN.

Application check support

You can specify that logon exit B02UXSIN is to check whether a user is authorized in RACF to use an application. As a minimum, the user must have access READ. To enable the application check, please do the following:

1. In RACF, define the application in the RACF APPL facility class. The application name consists of the 3-character product ID (for example, B88, B92, B93, etc.) plus the 3-character application ID (for example, WHD or BWE). For example, the application name checked for _beta access easy is B88WHD. For _beta view, it is B92BWE, B93BWE, etc.
2. Delete string ***APPL** from the corresponding statements in exit B02UXSIN, i.e. activate the assembler commands in these statements.
3. Recompile the exit into the BETA.APFLOAD used by the product (G#02XSIN in BSA.CNTL or BSTUXASM in BSA.SAMPLIB).

Passphrase support

You can specify that the logon exit B02UXSIN is to check passphrases as well as passwords. A passphrase can consist of 9 to 100 characters, and can be defined in addition to a password. To enable a check on passphrases, do the following:

1. Delete string ***PASS** from the corresponding statements in the exit, i.e. activate the assembler commands in these statements.
2. Recompile the exit into the BETA.APFLOAD used by the product (G#02XSIN in BSA.CNTL or BSTUXASM in BSA.SAMPLIB).

Please note that you can change an old password to a new password, or an old passphrase to a new passphrase, but you cannot change a password to a passphrase or a passphrase to a password.

Mixed case and special characters

The server logon exit B02UXSIN automatically checks the password handling defined in RACF. In accordance with the RACF settings, it activates/deactivates support of mixed-case passwords and special characters in passwords in the exit.

If you are using another security system, adjust the server logon exit manually if you need support for mixed-case passwords and special characters. For more information, see the comments in the source code of B02UXSIN, which is provided in the BSA.SAMPLIB.

For your changes to take effect, recompile the exit into the BETA.APFLOAD used by the product (G#02XSIN in BSA.CNTL or BSTUXASM in BSA.SAMPLIB).

Warning if exit is not installed

If the BSA TCP/IP server logon exit has not been installed, a warning message (9283W or 9284W or 8532W) is written to the operator console, but the user can continue working.

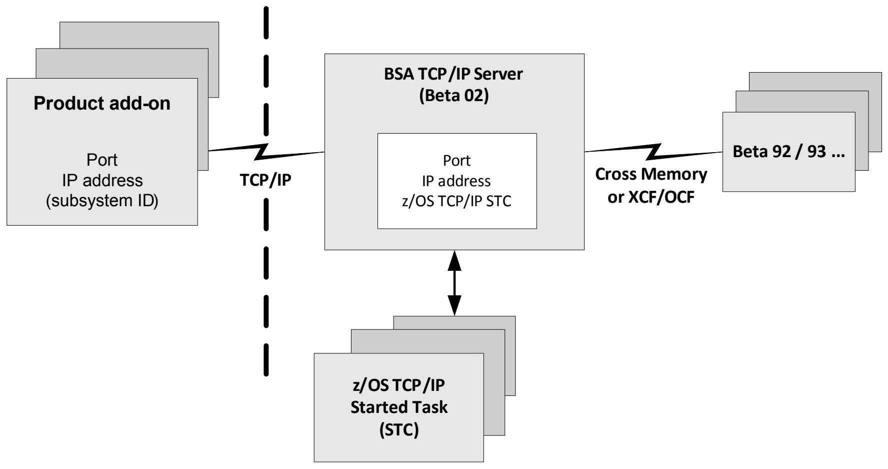
Product-specific exits

The BSA TCP/IP server logon exit can also call an additional product-specific logon exit. The name of the product-specific logon exit is specified in the LST parameter B*nn*_TCPIP_LOGON_EXIT or B*nn*_TCPIP_LOGON_EXIT_*app*.

BSA TCP/IP server as a standalone STC (Beta 02)

Overview

The BSA TCP/IP server can be activated as a standalone started task in Beta 02 (the installed Beta product level is not relevant).



Procedure

To set up Beta 02 as a standalone started task, you will need to do the following:

- Create JCL for the Beta 02 STC

  A sample procedure for the standalone Beta 02 started task can be found in member BETA02 in the BSA.SAMPLIB.

- Customize LST member B02LST*xx*

  Code the appropriate LST parameters for TCP/IP support in the B02LSTxx member of the Beta 02 started task. A sample can be found in member B02LST00 in the BSA.SAMPLIB. If a product is installed on a different LPAR, additional OCF definitions are required.

- Assemble and link the BSA TCP/IP server logon exit B02UXSIN

  For more information on B02UXSIN, see "BSA TCP/IP server logon exit (B02UXSIN)" on page 142.

- Create a member B02SSI*xx* (parameters for initializing the subsystem)

  An sample member with SSI parameters can be found under the name B02SSI00 in the BSA.SAMPLIB (see "SSI parameters" on page 145).

- Define the subsystem ID and run BST01ARI

Required libraries

The Beta 02 started task needs access to the following. Concatenate the required libraries in the STEPLIB statement if they are not included in the linklist.

- BSA.LOAD
- BSA TCP/IP server logon exit B02UXSIN (typically in the BETA.APFLOAD)
- BETA*nn*.LOAD of **each product** that is to be serviced by the Beta 02 STC

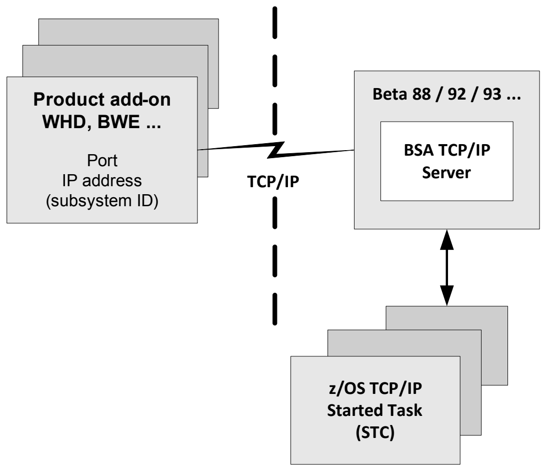
SSI parameters

Keyword	Value	Description	Option
UXSRT	BST00STH	Security router The value of the parameter must always be BST00STH.	required
UXSIN	*name*	Name of the BSA TCP/IP server logon exit We recommend that you do **not** change the name B02UXSIN. Do not specify a name if **no check** is to be performed when a user logs on. The user might be able to work with the system anyway, depending on the specific product and/or add-on.	optional

BSA TCP/IP server in a Beta product STC

Overview

The TCP/IP can be activated in a Beta product started task (as a master or slave STC):



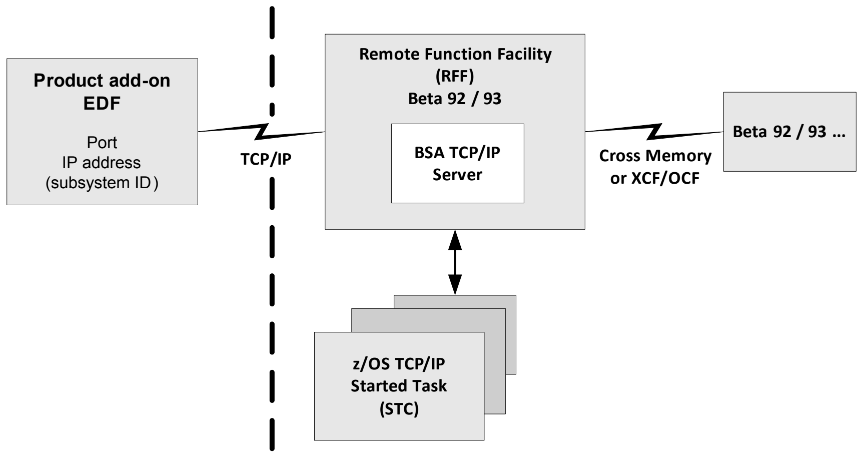
Activating the server

The BSA TCP/IP server with all its components is activated automatically provided that the Beta product concerned supports this functionality.

BSA TCP/IP server as an RFF job for EDF

Overview

The BSA TCP/IP server can be activated as a Beta product RFF job for the add-on EDF.



Prerequisites for EDF

To use the server for the add-on EDF 92/93, three prerequisites must be fulfilled:

1. The server functionality must have been implemented beforehand in the start-up sequence of the product (SICA).
2. EDF must run as a Remote Function Facility (RFF) system.
3. When BST01RFF is called up, B93TCPS or B92TCPS must be entered in the parameter PGM=.

When the server is used for EDF, by default the server-client program BxxTCL used previously will also be used by the BSA TCP/IP server.

Port keyword

B*nn*_TCPIP_PORT_RDR must be specified. All other port keywords are ignored.

SFF client groups for a TCP/IP port

Introduction

The LST parameter B*nn*_TCPIP_MAX_CLIENT_GROUPS enables you to enhance the throughput of a TCP/IP port by defining SFF client groups for the port. As a rule, one SFF client group is sufficient for normal product TCP/IP clients. However, in the event of heavy workloads on the various applications, the following values are recommended:

- Web applications (e.g. BWE) – maximum of 5 SFF client groups
- Agents (e.g. OSY) – maximum of 10 SFF client groups.

Client groups as solution

If you are using the BSA TCP/IP server, it is possible to create up to 26 SFF client groups (TCL groups) for a TCP/IP port. This not only takes the z/OS dispatcher control function into account, but also makes very efficient use of the internal SFF dispatcher control function, enabling throughput to be increased.

Defining client groups

A maximum of 26 client groups can be defined using LST parameter B*nn*_TCPIP_MAX_CLIENT_GROUPS=*xx* (valid for all global ports) or B*nn*_TCPIP_MAX_CLIENT_GROUPS_*app*=*xx*, (valid for all ports for the specified application *app*), where *xx* specifies the number of SFF client groups to be generated for the port. If more than 26 client groups are specified, the parameter is automatically reset to 26. The default is 1. When the product STC is started, the number of client groups specified here are defined.

Naming conventions for client groups

SFF client groups can be displayed by this operator command:

```
F stcname,TL
```

The name of the SFF client group is determined by the port and the application, and by the number of client groups that have been defined, for example:

No. of client groups	Name components	Description	Example
1 only	*app_port*	*app* = name of the application, e.g. OSY, BWE) *port* = Application port as HEX (e.g. *port* = 64000 = FA00 HEX)	OSY_FA00
More than 1	*appportb*	*app* = name of the application, e.g. OSY, BWE) *port* = Application port as HEX (e.g. *port* = 64000 = FA00 HEX) *b* = Consecutive number represented by the letters A to Z.	OSYFA00A

Limitations

- SFF Client groups can only be defined for the BSA TCP/IP server. They are **not** supported by the BSA Communication Integrator.
- SFF client groups can **not** be modified dynamically using the BSA Service Manager during runtime of the STC.
- If values for client groups are set too high, an increase in parallel processing could also lead to an increase in CPU usage.

Operator commands for the TCP/IP server

Overview

Various operator MODIFY commands are available for controlling the TCP/IP server.

The long and the short version of each command are shown in the following table.

Command (long)	Command (short)	Purpose
`ACT TCP`	`A TCP`	Start the server for a particular port
`INACT TCP`	`IN TCP`	Stop the server for a particular port
`REFRESH TCP`	`REF TCP` or `R TCP`	Stop and restart the server
`DISPLAY TCP`	`D TCP`	Display the status of ports and users
`CANCEL TCP`	`C TCP`	Cancel the users logged onto the port

ACT: Starting the server for a special port

Use the ACT command ACT to start (activate or reactivate) servers for particular ports, e.g. when the retry cycle for a port has elapsed. You cannot start servers for ports that are not available in the LST member.

Command keywords that are not entered in the command will be taken from the product LST member, or the default values for the LST keywords will be used.

If none of the parameters described is available in the LST member, the parameters marked as required in the list of LST parameters must be specified in the command.

Command	Keyword		

`ACT TCP`
-OR-
`A TCP`
	`PORT(`*`portnumber`*`)`
	`PROD(`*`product`*`)`
	`TYPE(`*`porttype`*`)`
		-OR-
		-OR-
			`P(`*`portnumber`*`)`
			`PR(`*`product`*`)`

	Replace...		**With...**
	portnumber	—	The number of the port you want to activate. If you enter *, the ports entered in the LST parameters of the product the server is running under will be started. If other ports are to be started, * must also be entered for parameter PROD.
	product	—	2-digit product identifier (for example: 93). Masks are allowed (for example: *).
	porttype	—	The name of the port to be activated, for example, BWE, OSY, BSA or * (where BSA and * are identical).

Examples

```
F betastc,ACT TCP,P(4711) PR(*) TYPE(*)
F betastc,ACT TCP,PORT(*) PR(*) TYPE(*)
F betastc,A TCP,P(4711) PROD(93) TYPE(BWE)
```

INACT: Stopping the server

Use the INACT command to stop (deactivate) a server for a specific port. All users logged onto the system via specific port will then be canceled.

Command	Keyword		
`INACT TCP`	`PORT(`*`portnumber`*`)`	-OR-	`P(`*`portnumber`*`)`
-OR-	`PROD(`*`product`*`)`	-OR-	`PR(`*`product`*`)`
`IN TCP`	`TYPE(`*`porttype`*`)`		
	Replace...		**With...**
	portnumber	—	The number of the port you want to stop.
	product	—	2-digit product identifier (for example: 93).
	porttype	—	The name of the port to be activated, for example, BWE, OSY, BSA or * (where BSA and * are identical).

Examples

```
F betastc,INACT TCP,P(4711) PR(93) TYPE(BWE)
F betastc,IN TCP,P(4711) PROD(92) TYPE(OSY)
```

REFRESH: Stopping and restarting the server

Use the REFRESH command to automatically stop and restart the server.

All users are canceled.

Command	Keyword		
`REFRESH TCP`	`PORT(`*`portnumber`*`)`	-OR-	`P(`*`portnumber`*`)`
-OR-	`PROD(`*`product`*`)`	-OR-	`PR(`*`product`*`)`
`REFR TCP`	`TYPE(`*`porttype`*`)`		
	Replace...		**With...**
	portnumber	—	The number of the port you want to refresh.
	product	—	2-digit product identifier (for example: 93).
	porttype	—	The name of the port to be activated, for example, BWE, OSY, or BSA.

DISPLAY: Displaying server and user status

Use the DISPLAY command to display the status of all the ports activated for the server, and to display all the users who are logged onto the server through these ports.

When user status is displayed, a message informing you about the number of users logged onto the server will be written to the console, and detailed user information will be written to the SYSLOG of the started task (STC).

Command	**Keyword**		
`DISPLAY TCP`	`PORT(`*`portnumber`*`)`	-OR-	`P(`*`portnumber`*`)`
-OR-	`PROD(`*`product`*`)`	-OR-	`PR(`*`product`*`)`
`D TCP`	`[ USER(`*`username`*`) ]`	-OR-	`[ U(`*`username`*`) ]`
	`TYPE(`*`porttype`*`)`		
	`[` *`connections`* `]`		
	Replace...		**With...**
	portnumber	—	The number of the port you want to display. When you enter *, all the ports entered in the LST parameters of the product the server is running under will be displayed.
	username	—	The name of the user who is to be displayed. If you enter *, all users activated for this port will be displayed. Masks are allowed, for example BER*. If you want to display only the users logged onto one product, use the parameter PROD.
	product	—	2-digit product identifier (for example: 93). Masks are allowed (for example: *****).
	porttype	—	The name of the port to be displayed, for example, BWE, OSY, or BSA.
	connections	—	CONNECT (or CONN or CO) to display all connections that are currently active CONNALL to display all connections (active and active) that have occurred during the runtime of the BSA TCP/IP server. CONNENC to show all connections (active and active) where the encryption method requested by the client does not match the encryption method of the server **Note**: The requested information must be available (B*nn*_TCPIP_CONNECT_INFO = YES; the default is: YES).

Examples

```
F betastc,DISPLAY TCP,U(*) PR(*) P(*) TYPE(*)
F betastc,D TCP,U(BER*) PORT(4711) PR(*) TYPE(BWE)
F betastc,D TCP,P(*) PR(*) TYPE(*) CONNENC
```

CANCEL: Canceling users

Use the CANCEL command to cancel users who are logged onto the server.

Command	Keyword		
`CANCEL TCP`	`PORT(`*portnumber*`)`	-OR-	`P(`*portnumber*`)`
-OR-	`PROD(`*product*`)`	-OR-	`PR(`*product*`)`
`C TCP`	`USER(`*username*`)`	-OR-	`U(`*username*`)`
	`TYPE(`*porttype*`)`		
	Replace...		**With...**
	portnumber	—	The number of the port where you want to cancel users.
	product	—	2-digit product identifier (for example: 93).
	username	—	The name of the user who is to be displayed. If you enter *, all users activated for this port will be displayed. Masks are allowed, for example BER*. If you want to display only the users logged onto one product, use the parameter PROD.
	porttype	—	The name of the port where the user is to be canceled, for example, BWE, OSY, or EDF.

Examples

```
F betastc,CANCEL TCP,U(*) PORT(4711) PR(93) TYPE(BWE)
F betastc,C TCP,U(BER*) PORT(4711) PR(92) TYPE(OSY)
```

LST parameters for the BSA TCP/IP server

Overview

This section describes the LST parameters that are available for controlling the BSA TCP/IP server.

***_app* suffix**

Some applications (add-ons) may need to be controlled differently from other applications. This is why you can define application ports, which can only be used by the specified application of one product. Use a global port instead if you want different applications of one or more products to share the same port.

A keyword with the ***_app*** suffix defines values for an application port, where ***app*** is the 3-character identifier of the application (add-on). A keyword without the ***_app*** suffix is generally valid for all applications of the product or defines values for a global port.

Most keywords can be coded with and without the ***_app*** suffix. Keywords with the prefix **BSA_** are valid for all applications and products.

Examples:

```
B93_TCPIP_PORT
B93_TCPIP_PORT_BWE
B92_TCPIP_PORT_OSY
BSA_TCPIP_TRACE
```

Examples

Examples on how to use the LST parameters for the BSA TCP/IP server can be found in the B02LST00 member in the BSA.SAMPLIB.

Changing parameters dynamically

A subset of the LST parameters can be set or changed dynamically via the BSA Service Manager.

The BSA Service Manager panel "Display Modifiable Parameter Keywords/Values" (Option **D.S.1.2**) shows a complete list of the parameters that you can change dynamically in your subsystem. For more information on using this option, see the *BSA Service Manager Manual*.

Note: When you set or change parameters using the BSA Service Manager, these changes are temporary and will no longer be valid after restarting the started task.

PORT LST parameter

Following is a description of the PORT LST parameter. For a description of the other LST parameters of the BSA TCP/IP server, see "Parameter descriptions" on page 157.

Syntax

```
                                              Bnn_TCPIP_PORT[_app] =
port,ipa[<resolve>],tcpname[,[clnt],[mssid],[rssid],[keepalive]]
```

Values are coded as a comma-separated list of positional parameters. The port number (*port*), the IP address (*ipa*), and the name of the TCP/IP started task (*tcpname*) are required. The other parameters are optional.

Parameter	Description	Opt./Req.
port	Listening port (max. 5 digits)	required
ipa[<*resolve*>]	Bind address (numeric or symbolic) Use the following notations if you want to specify an all-zeros address to bind to all available interfaces: 0.0.0.0 All available IPv4 addresses; accepts only connections via IPv4 protocol. IPA_ANY4 Same as **0.0.0.0** :: All available IPv4 and IPv6 addresses; accepts connections via IPv4 and IPv6 protocol. IPA_ANY6 Same as **::** Symbolic addresses (max. 255 characters) are resolved to numeric addresses (IPv6 and/or IPv4) via DNS. The resolution sequence is determined by the DNS server. By default, IPv6 addresses have precedence over IPv4 addresses. You can use the ***<resolve>*** parameter to control DNS resolution if ***ipa*** is a symbolic address: *ipa* The TCP/IP server uses all addresses from DNS resolution and binds to the first reachable IPv6 address and to the first reachable IPv4 address. This is the default, which is adequate in most situations. *ipa*<IPV6> The TCP/IP server uses all IPv6 addresses from DNS resolution and binds to the first reachable address. *ipa*<IPV4> The TCP/IP server uses all IPv4 addresses from DNS resolution and binds to the first reachable address. *ipa*<IPV6:ALL> The TCP/IP server uses all IPv6 addresses from DNS resolution and binds to all reachable addresses. *ipa*<IPV4:ALL> The TCP/IP server uses all IPv4 addresses from DNS resolution and binds to all reachable addresses. *ipa*<ALL> The TCP/IP server uses all IPv6 and IPv4 addresses from DNS resolution and binds to all reachable addresses. The ***<resolve>*** parameter has no effect if ***ipa*** is a numeric address.	required
tcpname	Name of the TCP/IP started task (TCP/IP stack) on the z/OS system (max. 8 characters)	required

Parameter	Description	Opt./Req.
clnt	This parameter should only be used in exceptional situations. Specify only if asked to do so by Beta Systems support: Program name of the server client (max. 8 characters)	optional
mssid	Use this parameter if the BSA TCP/IP server runs as a standalone Beta 02 STC and ports are defined for _beta doc|z (B93_TCPIP_PORT[_*app*]) or _beta doc|z plus (B97_TCPIP_PORT[_*app*]): Specify the *ssid* (max. 4 digits) of the subsystem if it should receive copies of TCP/IP messages from Beta 02 STC. The receiving subsystem can store these messages in its message database.	optional
rssid	This parameter is necessary for the keyword B*nn*_TCPIP_PORT_EDF if the BSA TCP/IP server runs as a standalone Beta 02 STC: The subsystem ID (max. 4 digits) where requests are to be sent	optional
keepalive	Interval in seconds that is used to send keepalive probes When no data flows, TCP keepalive processing periodically sends packets over the TCP connection to prevent the connection from being reset. The default is okay for most applications, and there is no need to change it. Beta Systems support may ask you to change this value if you are experiencing TCP connection reset problems. Several LST parameters can be used to specify the interval in seconds that is used to send keepalive probes. When the keepalive value is set for a specific port, the following hierarchy applies (from highest to lowest): 1. Value specified in B*nn*_TCPIP_PORT[_*app*] 2. Value specified in B*nn*_TCPIP_KEEPALIVE_TIME_*port* 3. Value specified in B*nn*_TCPIP_KEEPALIVE_TIME Valid values: 0..2147460 (default is 5 seconds; 0 deactivates keepalive) It is possible to modify the keepalive value of each active port dynamically with the help of the BSA Service Manager (option **D.S.4.2.2**, Action: Display; Line command **L** (Limit)). The modification takes effect when the first new connection is established after the value has been modified. For detailed information on the handling of the TCP_KEEPALIVE value, see also *z/OS Communications Server: IP Programmer's Guide and Reference*.	optional

Parameter descriptions

Keyword	Value	Description	Opt./Req.	Default
B*nn*_TCPIP_BACKLOG	*nnnnnnnnn*	Defines how many pending connections the queue will hold. This parameter applies to all global and application ports defined for the product.	Optional	255
B*nn*_TCPIP_COMPRESS B*nn*_TCPIP_ COMPRESS_*app*	*method*	Specifies the compression method to be used This keyword excludes the use of the keyword B*nn*_TCPIP_CRYPT_EXIT. Only use this keyword if the product client supports compression. The following methods can be specified: STD Beta standard compression YES Same as **STD** NO No compression	Optional	NO
B*nn*_TCPIP_ENCRYPT B*nn*_TCPIP_ ENCRYPT_*app*	*method* [,*key*]	Specifies the encryption method to be used Optionally, you can also specify the key. Keys must consist of HEX values only (0123456789ABCDEF). If no key is specified, a key based on the indicated method will be generated automatically during each connect. Only use this keyword if the product client supports this encryption. The following methods can be specified: DES DES algorithm; key length = 16 TRD Triple DES algorithm, key length = 48 AES Advanced encryption standard, key length = 64 (encryption mode CBC; fallback to Triple DES if client does not support AES) BFS Blowfish algorithm, key length >=16 and <=128, divisible by 2 NO No encryption	Optional	NO

Keyword	**Value**	**Description**	**Opt./Req.**	**Default**
B*nn*_TCPIP_KEEPALIVE_TIME B*nn*_TCPIP_KEEPALIVE_TIME_*port*	0..2147460	Interval in seconds that is used to send keepalive probes For details on valid values and handling, see "PORT LST parameter" on page 154.	Optional	5
B*nn*_TCPIP_LOGON_EXIT B*nn*_TCPIP_LOGON_EXIT_*app*	*name*	Use only if supported by the product (see product documentation): Name of the product logon exit that is to be called by the default BSA TCP/IP server logon exit	Optional	*none*
B*nn*_TCPIP_MAX_CLIENT B*nn*_TCPIP_MAX_CLIENT_*app*	*nnnn*	Specifies the number of client connects/ logons that can be made in parallel. Once this number has been reached, no more connects will be accepted until an existing one is closed. The BSA TCP/IP server checks this cyclically after the period of time specified by B*nn*_TCPIP_CLIENT_WAIT.	Optional	1000
B*nn*_TCPIP_MAX_CLIENT_GROUPS B*nn*_TCPIP_MAX_CLIENT_GROUPS_*app*	*nn*	*nn* specifies the number of SFF client groups to be generated for the port. If more than 26 client groups are specified, the parameter is automatically reset to 26. When the product STC is started, the number of client groups specified here are defined.	Optional	1

Keyword	Value	Description	Opt./Req.	Default
B*nn*_TCPIP_MAX_ CLIENT_REJECT B*nn*_TCPIP_MAX_ CLIENT_REJECT_*app*	YES | NO	This parameter controls how the BSA TCP/IP server handles incoming TCP/IP connect attempts when the maximum number of active clients has been reached for a port (see B*nn*_TCPIP_MAX_ CLIENT[_*app*]). If **NO**: The client waits for a connection. The BSA TCP/IP server checks for closed connections at the specified interval. If **YES**: The BSA TCP/IP server rejects the connection attempt without waiting. This is indicated by messages 9280W and 9281W. In case of EDF, the connection is refused immediately after the messages are output. In case of other applications, the BSA TCP/IP server completes the protocol handshake with the requestor (client), sends specific terminating information according to an internal protocol, and then closes the connection. There is no message indicating that the BSA TCP/IP server is ready to accept connects again. This handling applies to all applications in case of a global port. Code the application identifier _*app* in the keyword if this handling should apply only to a specific application port.	Optional	NO
B*nn*_TCPIP_CLIENT_WAIT B*nn*_TCPIP_CLIENT_ WAIT_*app*	*nnnn*	Specifies the number of seconds that the BSA TCP/IP server waits before checking whether the number of clients specified by B*nn*_TCPIP_MAX_CLIENT is no longer exceeded.	Optional	5
B*nn*_TCPIP_PORT B*nn*_TCPIP_PORT_*app*		See "PORT LST parameter" on page 154.	Required	*none*

Keyword	**Value**	**Description**	**Opt./Req.**	**Default**
B*nn*_TCPIP_RETRY_ INTERVAL	*secs*,*num*	The time and the number of the retry intervals (max. 9 digits each). If the task is not active or the connection is lost, an attempt will be made to automatically reconnect the server to the TCP/IP started task after *secs* has elapsed. This will be repeated *num* times. *secs* = the time in seconds the system is to wait before a reconnect is attempted. *num* = the number of retry intervals. This number specifies how often a reconnect between a server and the z/OS TCP/IP stack or STC is attempted. If no reconnect to the system can be made within the specified number of retry intervals, the server will be terminated. You can reactivate the server by using the appropriate MODIFY command. **0** means the number of retries is unlimited. This parameter applies to all global and application ports defined for the product.	Optional	60,0
B*nn*_TCPIP_ROUTER B*nn*_TCPIP_ROUTER_*app*	YES | NO	Select NO to attach the client directly via the BSA TCP/IP server and not via the TCP/IP router by means of bss_srqe.	Optional	YES
BSA_TCPIP_SEND_ASY	YES | NO	In synchronous mode (NO), the entire SFF group that includes the sending function pauses until the send process has finished successfully. In asynchronous mode (YES), only the sending function within the SFF group pauses until the send process has finished successfully. Other functions within this SFF group can continue working in parallel.	Optional	YES
B*nn*_TCPIP_SESS_ TIME_LIMIT	*nnnnnnnnn*	Enter the maximum period of time (time limit) in minutes a user can remain inactive before being timed out. When the time is exceeded, the user will be disconnected. Disconnected users must log onto the z/OS system again if they want to proceed. This parameter applies to all global and application ports defined for the product.	Optional	60

Keyword	Value	Description	Opt./Req.	Default
B*nn*_TCPIP_TIMEOUT	*nnnnnnnnn*	The space of time (max. 9 digits) in seconds after the program will cancel a session due to system inactivity. If the time specified here elapses before any event is detected, the TCP/IP SELECT command will return the control. The timeout value set via this parameter applies to all global and application ports defined for the product.	Optional	60
BSA_TCPIP_TRACE	YES | NO | SHO | ERR | SHORT | ERROR	If YES, all server and server client activities are logged, i.e. all requests. Specifying the value SHO or SHORT or the value ERR or ERROR for this parameter reduces the amount of information that is logged. Set this parameter to YES or SHO or ERR only if asked to do so by Beta Systems support.	Optional	NO
BSA_TCPIP_TRACE_LST	YES | NO	If YES, all the LST keywords that are processed while the server is starting are logged. Set this parameter to YES only if asked to do so by Beta Systems support.	Optional	NO
BSA_TCPIP_TRACE_ INTERN	YES | NO	If YES, all TCP/IP requests are logged. Set this parameter to YES only if asked to do so by Beta Systems support.	Optional	NO
BSA_TCPIP_TRACE_ SNDRCV	YES | NO	This parameter has an effect only if BSA_TCPIP_TRACE_INTERN = YES: YES All TCP/IP SEND or RECEIVE requests are logged. NO Only the first SEND and RECEIVE requests are logged. Set this parameter to NO only if asked to do so by Beta Systems support.	Optional	YES
BSA_TCPIP_TRACE_BUF	YES | NO	If YES, all buffer data from SEND or RECEIVE requests are logged in dump format. Set this parameter to YES only if asked to do so by Beta Systems support.	Optional	NO
B*nn*_WLM_SUBSYS_TYPE B*nn*_WLM_SUBSYS_ NAME		See "LST parameters for ZIF" on page 213		

Scope of keywords with [_app]

Please note the difference in scope when the following two groups of keywords are used with and without the ***_app*** suffix:

Keyword	**Scope**
B*nn*_TCPIP_LOGON_EXIT[_*app*] B*nn*_TCPIP_COMPRESS[_*app*] B*nn*_TCPIP_ENCRYPT[_*app*]	The keywords in this group define settings for applications; settings defined for an application are independent of the type of port being used (global or application): • The keywords **with *_app*** suffix apply to the corresponding **application** of Beta *nn*. • The keywords **without *_app*** suffix apply to **all applications** of Beta *nn*.
B*nn*_TCPIP_CLIENT_WAIT[_*app*] B*nn*_TCPIP_MAX_CLIENT[_*app*] B*nn*_TCPIP_MAX_CLIENT_REJECT[_*app*] B*nn*_TCPIP_MAX_CLIENT_GROUPS[_*app*]	The keywords in this group define settings for ports; settings defined for a global port are independent of the type of application being used: • The keywords **with *_app*** suffix apply to the corresponding **application ports** defined for Beta *nn*. • The keywords **without *_app*** suffix apply to the **global ports** defined for Beta *nn*.

Securing TCP/IP connections with AT-TLS

Overview

The BSA TCP/IP server relies on AT-TLS to secure its TCP connections from and to client partners.

AT-TLS policy is read, parsed, and installed into the TCP/IP stack by the z/OS Communications Server Policy Agent (PAGENT), which implements policy-based networking for the z/OS environment (protocols, cipher suites, etc.).

Hand-coding or z/OSMF

There are two approaches to configuring AT-TLS and Policy Agent on z/OS:

- You can use the z/OS Management Facility (z/OSMF) Configuration Assistant for z/OS Communications Server. The Configuration Assistant is a graphical user interface (GUI)-based tool that simplifies the setup, configuration, and deployment of z/OS Communications Server policy-based technologies, including AT-TLS.
- You can hand-code all of the necessary job control language (JCL), RACF directives, configuration files, and policy files. To provide the reader greater insight into the Policy Agent and AT-TLS configuration, this is the approach we follow in the remainder of this description.

Requirements (summary)

Following is a summary of the actions and definitions that are required for a complete implementation of AT-TLS between the BSA TCP/IP server and the client partner. These steps are described in more detail in the following sections.

1. Configuring Policy Agent (PAGENT) as a started task in z/OS
2. Defining the security authorization for Policy Agent
3. Defining the Policy Agent configuration files
4. Configuring AT-TLS
5. Defining the AT-TLS policy rules
6. Creating and configuring digital certificates in RACF
7. Configuring a secure connection for the TCP/IP server and the client partner

Configuring Policy Agent as a started task in z/OS

Overview

Policy Agent (PAGENT) runs as a UNIX processor as a z/OS started task. In our example, we use the z/OS started task procedure to start Policy Agent.

STC procedure

Sample JCL for the Policy Agent started task can be found in member PAGENT in the BSA.SAMPLIB.

```
+----------------------------------------------------------------------+
|//PAGENT   PROC                                                       |
|//******************************************************************** |
|//*                                                                 * |
|//*  PAGENT STARTED PROCEDURE                                       * |
|//*                                                                 * |
|//*  SAMPLE-PROCEDURE                                               * |
|//*                                                                 * |
|//*                                                                 * |
|//******************************************************************** |
|//PAGENT   EXEC PGM=PAGENT,REGION=0K,TIME=NOLIMIT,                    |
|//       PARM='ENVAR("_CEE_ENVFILE=DD:STDENV")/'                       |
|//*                                                                   |
|//* FOR INFORMATION ON THE ENVIRONMENT VARIABLES, REFER TO THE        |
|//* IP CONFIGURATION GUIDE.                                           |
|//*                                                                   |
|//*** EXAMPLES FOR SPECIFYING ENVIRONMENT VARIABLES                   |
|//*                                                                   |
|//* SAMPLE UNIX FILE CONTAINING ENVIRONMENT VARIABLES:                |
|//*   STDENV   DD PATH='/etc/pagent.env',PATHOPTS=(ORDONLY)          |
|//*                                                                   |
|//* SAMPLE z/OS DATA SET CONTAINING ENVIRONMENT VARIABLES:            |
|//*   STDENV   DD DSN=BETA.BSA.SAMPLIB(PAGNTENV),DISP=SHR            |
|//*                                                                   |
|//STDENV   DD DSN=BETA.BSA.SAMPLIB(PAGNTENV),DISP=SHR                |
|//*                                                                   |
|//SYSPRINT DD SYSOUT=*                                                |
|//SYSOUT   DD SYSOUT=*                                                |
|//*                                                                   |
|//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)          |
+----------------------------------------------------------------------+
```

Configuration file

DD STDENV defines the name of the configuration file for Policy Agent. The configuration file can be a z/OS dataset or a Unix file.

In our example, we use a member in a z/OS dataset. A sample configuration file can be found in member PAGNTENV in the BSA.SAMPLIB.

```
LIBPATH=/lib:/usr/lib:usr/lpp/ldapclient/lib:.
PAGENT_CONFIG_FILE=//'BETA.BSA.IP.PARMLIB(PAGNTCFG)'
PAGENT_LOG_FILE=/tmp/pagent.log
PAGENT_LOG_FILE_CONTROL=300,3
_BPXK_SETIBMOPT_TRANSPORT=TCPIP
```

Important:

- Parameter names and values are case-sensitive.
- Line numbering must be turned off when editing this member.

Defining the security authorization for Policy Agent

Overview

The policies managed by Policy Agent can affect system operation significantly. Therefore, you need to restrict the list of z/OS user IDs under which Policy Agent is allowed to run. To do this, you need to define certain resources and controls in the system's security manager product, such as RACF.

Security definitions

To set up the Policy Agent's security definitions to RACF:

1. Define the PAGENT user ID.
2. Define the PAGENT started task to RACF.
3. Associate the PAGENT user ID with the PAGENT started task.
4. Give authorized users access to manage the PAGENT started task.

RACF command example

Following is an example of sequence of RACF commands that set up security definitions for Policy Agent.

1. Define a user ID (for example, PAGENT) for the PAGENT started task:

   ```
   ADDUSER PAGENT DFLTGRP(OMVSGRP) OMVS(UID(0) HOME('/'))
   ```

2. Define the PAGENT started task to RACF:

   ```
   RDEFINE STARTED PAGENT.*  DFLTGRP(OMVSGRP) -
           STDATA(USER(PAGENT) GROUP(OMVSGRP)
   SETROPTS RACLIST(STARTED) REFRESH
   SETROPTS GENERIC(STARTED) REFRESH
   ```

3. If you want to log messages through SYSLOGD, define a profile for SYSLOGD:

   ```
   RDEFINE STARTED SYSLOGD.*
   ```

4. Give authorized users access to manage the PAGENT started task:

   ```
   PERMIT MVS.SERVMGR.PAGENT CLASS(OPERCMDS) ACCESS(CONTROL) -
           ID(PAGENT)
   SETROPTS RACLIST(OPERCMDS) REFRESH
   SETROPTS GENERIC(OPERCMDS) REFRESH
   ```

5. Optional: Restrict access to the **pasearch** Unix command.

   ```
   PERMIT EZB.PAGENT.SYSTEMNAME.TCPIPNAME.* CLASS(SERVAUTH) -
           ID(userid) ACCESS(READ)
   SETROPTS RACLIST(SERVAUTH) REFRESH
   SETROPTS GENERIC(SERVAUTH) REFRESH
   ```

A sample job with these security definitions can be found in member PAGNTSEC in the BSA.SAMPLIB.

Defining the Policy Agent configuration files

Overview

The main configuration file is used to configure certain operational characteristics of Policy Agent.

Statements

The main configuration file contains the following statements:

- The TcpImage statement specifies a TCP/IP image (a TCP/IP stack) and its associated image-specific configuration file.
- The LogLevel statement controls the amount of information logged by Policy Agent.
- The TTLSConfig statement identifies the AT-TLS policy file for this TCP/IP stack.

Example

A sample configuration file can be found in member PAGNTCFG in the BSA.SAMPLIB.

```
# POLICY AGENT CONFIGURATION FILE
#
# LOGLEVEL STATEMENT
# SYSERR, OBJERR, PROTERR, AND WARNING MESSAGES ARE LOGGED.
#
LogLevel 15
#
# AT-TLS policy for the TCPIP image:
#
TcpImage TCPIP  FLUSH PURGE 3600
TTLSConfig  //'BETA.BSA.IP.PARMLIB(TTLSCONF)'
```

Important:

- Parameter names and values are case-sensitive.
- Line numbering must be turned off when editing this member.

Configuring AT-TLS

Overview

AT-TLS support can be enabled by the specifying the TTLS parameter on the TCPCONFIG statement in the TCP/IP profile dataset.

Recommended alternative: You can also activate/deactivate AT-TLS support dynamically.

Dynamic activation/ deactivation

The BSA.SAMPLIB includes sample members for the dynamic activation/deactivation of TTLS.

Activating TTLS

Member TTLSON is used for activating TTLS. This member contains the following:

```
TCPCONFIG TTLS
```

To activate TTLS, enter the following operator command on the console:

```
V TCPIP,tcpname,O,DSN=BSA.SAMPLIB(TTLSON)
```

Deactivating TTLS

Member TTLSOFF is used for deactivating TTLS. This member contains the following:

```
TCPCONFIG NOTTLS
```

To deactivate TTLS, enter the following operator command on the console:

```
V TCPIP,tcpname,O,DSN=BSA.SAMPLIB(TTLSOFF)
```

Sample commands

You can find sample commands in member TTLSCMD in the BSA.SAMPLIB.

RACF definitions for AT-TLS initialization

When AT-TLS is started during TCP/IP stack initialization, there might be a delay between the time that the stack comes up and when Policy Agent successfully installs policy information into the stack. This situation can leave a window of time where connections that are intended to be protected by AT-TLS can be established without that protection. Additional RACF definitions for AT-TLS initialization can be used to prevent access from the network before Policy Agent has been properly started and activated.

With an appropriate RACF profile in place, you can control which applications are allowed to establish TCP connections before Policy Agent is started. Give READ access to this profile to all tasks that need access to TCP/IP before Policy Agent is started and has terminated its initialization. All other applications are forced to wait until Policy Agent has initialized and its policies have been installed into the stack before connections are enabled.

Initialization access control for AT-TLS:

```
RDEFINE SERVAUTH EZB.INITSTACK.sysname.TCPIP UACC(NONE)
PERMIT EZB.INITSTACK.sysname.TCPIP CLASS(SERVAUTH) -
       ID(OMVSKERN) ACCESS(READ)
PERMIT EZB.INITSTACK.sysname.TCPIP CLASS(SERVAUTH) -
       ID(PAGENT) ACCESS(READ)
PERMIT EZB.INITSTACK.sysname.TCPIP CLASS(SERVAUTH) -
       ID(SYSLOGD) ACCESS(READ)
SETROPTS GENERIC(SERVAUTH) REFRESH
SETROPTS RACLIST(SERVAUTH) REFRESH
```

A sample job can be found in member TTLSRAC in the BSA.SAMPLIB.

Tip: This type of profile is normally defined with UACC=NONE. To prevent that you lock yourself out of your z/OS system, we recommend that you define the profile with UACC=READ, and then define profiles with ACC=NONE to exclude the individual tasks that have to wait for AT-TLS. For more information, see *z/OS V2R1 Communications Server: IP Configuration Reference*.

Starting/Stopping PAGENT

Use the z/OS START command to start Policy Agent as a started task, for example:

```
S PAGENT
```

To shut down PAGENT (normally), use the z/OS STOP command, for example:

```
P PAGENT
```

Defining the AT-TLS policy rules

Overview

An AT-TLS policy configuration file contains AT-TLS rules that identify specific types of TCP traffic, along with the type of TLS/SSL to be applied to those connections. You can use a z/OS dataset or a Unix file.

Sample configuration for BSA TCP/IP server

In our example, we will use a z/OS dataset member. The following example describes AT-TLS policy parameters that are exclusively intended for use with the BSA TCP/IP server with TLS/SSL support. You can find this example in member TTLSCONF in the BSA.SAMPLIB.

Important:

- Parameter names and values are case-sensitive.
- Line numbering must be turned off when editing this member.
- The member must be edited with codepage 1047.
- BSA TCP/IP server ports must be defined with `ApplicationControlled on`.

```
###################################################################
#  BSA TCP/IP Server AT-TLS Support                               #
###################################################################
# Common Production Group that all Rules use
TTLSGroupAction BSA_Secure_GroupAct
{
 TTLSEnabled On
 Trace 15
}
###################################################################
#                                                                 #
#  BSA Specific Rules and Actions for BETA WebEnabler             #
#                                                                 #
###################################################################
TTLSRule                  BSA_Secure_Server_BWE_64193
{
 LocalPortRange           64193
#Jobname                  BETA02X
 Direction                Inbound
 TTLSGroupActionRef       BSA_Secure_GroupAct
 TTLSEnvironmentActionRef BSA_Secure_Server_Env
}

TTLSRule                  BSA_Secure_Server_BWE_64293
{
 LocalPortRange           64293
#Jobname                  BETA02X
 Direction                Inbound
 TTLSGroupActionRef       BSA_Secure_GroupAct
 TTLSEnvironmentActionRef BSA_Secure_Server_Auth_Env
}
```

(*continued*)

(continued)

```
# Environment Shared by all secure BSA server connections.
TTLSEnvironmentAction BSA_Secure_Server_Env
{
 HandshakeRole Server
 TTLSKeyRingParms
 {
   Keyring B02COMMON
 }
 TTLSEnvironmentAdvancedParms
 {
   ApplicationControlled On          # required
   HandshakeTimeout 10
   CertificateLabel   B02-PROD-CERT
   SecondaryMap  On   # include data connections
   SSLV2         Off  # Allow SSLv2 connections (Default is Off)
   SSLV3         Off  # Allow SSLv3 connections (Default is On )
   TLSV1         Off  # Allow TLSv1 connections (Default is On )
   TLSV1.1       Off  # Allow TLSv1.1 connections (Default is On )
   TLSV1.2       On   # Allow TLSv1.2 connections (Default is Off )
 }
}

# Environment Shared by all secure BSA server Clientauth connections.
TTLSEnvironmentAction BSA_Secure_Server_Auth_Env
{
 HandshakeRole ServerWithClientAuth
 TTLSKeyRingParms
 {
   Keyring B02COMMON
 }
 TTLSEnvironmentAdvancedParms
 {
   ApplicationControlled On     # required
   HandshakeTimeout 10
   CertificateLabel   B02-PROD-CERT
   SecondaryMap  On   # include data connections
   SSLV2         Off  # Allow SSLv2 connections (Default is Off)
   SSLV3         Off  # Allow SSLv3 connections (Default is On )
   TLSV1         Off  # Allow TLSv1 connections (Default is On )
   TLSV1.1       Off  # Allow TLSv1.1 connections (Default is On )
   TLSV1.2       On   # Allow TLSv1.2 connections (Default is Off )
 }
}
```

(continued)

```
(continued)
# Set of TLS Ciphers with Encryption
TTLSCipherParms RequireEncryption
{
  V3CIPHERSUITES            TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
  V3CIPHERSUITES            TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
  V3CIPHERSUITES            TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
  V3CIPHERSUITES            TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
  V3CIPHERSUITES            TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
  V3CIPHERSUITES            TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
  V3CIPHERSUITES            TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
  V3CIPHERSUITES            TLS_DHE_DSS_WITH_AES_256_GCM_SHA384
  V3CIPHERSUITES            TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
  V3CIPHERSUITES            TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
  V3CIPHERSUITES            TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
  V3CIPHERSUITES            TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
  V3CIPHERSUITES            TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
  V3CIPHERSUITES            TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
  V3CIPHERSUITES            TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
  V3CIPHERSUITES            TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
  V3CIPHERSUITES            TLS_RSA_WITH_AES_128_CBC_SHA256
  V3CIPHERSUITES            TLS_RSA_WITH_AES_256_CBC_SHA256
  V3CIPHERSUITES            TLS_RSA_WITH_AES_128_GCM_SHA256
  V3CIPHERSUITES            TLS_RSA_WITH_AES_256_GCM_SHA384
}
```

Note: Specify **Direction Outbound** if the z/OS client is to open a TLS connection to the Unix/Windows server, as in the following example:

```
# Host Client to PC Server ( with server handshake on the host side )
TTLSRule                  BSA_Secure_client_SPC
{
   RemotePortRange          63192
   Direction                Outbound
   TTLSGroupActionRef       BSA_Secure_GroupAct
   TTLSEnvironmentActionRef BSA_Secure_Client_SPC_Env
}
```

Activating AT-TLS policies

You can activate the AT-TLS policies dynamically with the help of the following operator command:

```
F PAGENT,UPDATE
```

Alternatively, you can also stop and then restart the PAGENT started task.

Verifying active policies

You can verify the active policies under OMVS with the help of the following command:

```
pasearch -t
```

Under TSO/ISPF, you can use the following commands for verification:

```
NETSTAT TTLS
```

`NETSTAT TTLS CO` *`number_from_pasearch command`* `DETAIL`

Creating and configuring digital certificates in RACF

Overview

Appropriate digital certificates are required to be able to use SSL/TLS.

Managing certificates

There are numerous ways to create and manage digital certificates and their related key pairs. Under z/OS, this includes the following options:

- z/OS Security Server RACF (and other SAF-compliant external security managers)
- The gskkyman tool that is included with System SSL
- z/OS Cryptographic Services PKI Services

In our example, we will use z/OS Security Server RACF.

Required actions

Following is a summary of the required actions:

1. Authorize the BSA TCP/IP server started task (for example BETA02W) to work with digital certificates.
2. Define a digital certificate together with a digital key ring. You can use a self-defining certificate or a CA-certificate.
3. Place the certificate into a dataset, transfer the dataset via ftp to the client PC, and then make the certificate available to the appropriate application on the PC (for example, Web Enabler). Transfer with ASCII conversion if the certificate is base64 encoded, otherwise transfer in binary mode.

The required commands can be found in a sample job in member TTLSCERT in the BETA.SAMPLIB.

You can also use the more detailed instructions for BSA CI as a model (see "Defining SSL authentication security" on page 192).

Note

Create the certificates etc. on the Unix/Windows server and make them available to the user ID of the z/OS client if the z/OS client is to open a TLS connection to the Unix/Windows server.

Securing a connection between BSA TCP/IP server and client partner

Server side

On the server side, follow the instructions for the BSA TCP/IP server for defining a global port or an application port.

Communication via this port will use SSL/TLS if the port is covered by appropriate rules in the AT-TLS policy rules configuration.

The user ID of the Beta product STC or Beta 02 STC must be set up with the required digital certificates.

Client side

On the client side, set up the client to use TLS/SSL when communicating with the z/OS host.

Provide trust if necessary by adding the certificate that you have transferred from the z/OS host to the truststore of the client.

For more information, see the product documentation of the client.

BSA Communication Integrator (BSA CI)

In this chapter

Topic **Page**

Introducing the BSA CI 175
System overview 177
BSA CI functions and features 180
Installing and customizing BSA CI 182
Creating JCL for the BSA CI started task 184
BSA CI ports 185
Defining a service port 186
Dynamic server control and automatic retry 187
Available runtime modes 188
Client authentication and authorization 191
Defining SSL authentication security 192
  Defining a security environment and certificates 193
  Creating and customizing a member for key ring and other parameters 200
Creating and customizing a member for port and other parameters 203
LST parameters for BSA CI 204
Compression and encryption for the BSA CI 210

Introducing the BSA CI

Overview

The BSA CI (Communication Integrator) enables cross-platform communication between different systems. For example, the BSA CI enables Web applications running under Windows or Unix to provide online access to systems running under z/OS.

The BSA CI is a concurrent server. This means that it can serve more than one client at a time by performing tasks at interleaved intervals. It can be used in common by all Beta Systems products and their add-ons, for example, Web Enabler.

Prerequisites

Before you start using the BSA CI, please refer to the relevant Beta Systems product documentation to make sure that the product concerned can use the BSA CI.

BSA CI architecture

The BSA CI consists of a request router, the concurrent BSA CI itself, and its own server client.

The BSA CI operates as a separate started task without a special subsystem ID.

What does BSA CI do?

The BSA CI can process requests from multiple Beta Systems products on multiple ports. Application ports can only be used by one product application. Global ports can be shared by different applications of the same product.

BSA CI supports the following core functions:

- Application ports and global ports
- Service ports
- SSL
- DNS (Domain Name System)
- Product exits
- Encryption and compression for ports not using SSL (NOSSL ports)
- JavaBSA support

Logon

The complete client authentication process (user logon) to the z/OS system is initiated by BSA CI and implemented by the BSA Service Manager in the product STC.

Communication

Communication between BSA CI and the product takes place via the BSA Cross Memory (CM) method or a BSA XCF connection, depending on the system definitions. All requests are sent directly to the product, without being routed via the BSA TCP/IP server.

SSL communication

BSA CI provides SSL communication, including components of a PKI (Public Key Infrastructure), between the product add-ons (applications) and the product server. It exchanges the relevant certificates while checking their authorization, and sends the verified requests to the appropriate Beta systems product for further processing. It then returns the results to the requestor.

SSL communication is generally based on System SSL (Secure Sockets Layer) components of the IBM z/OS operating system.

Service port functions

BSA CI supports service port functions that are initiated by the product server and sent to the product client application via SSL communication. In this case, internal communication between the product server and BSA CI takes place via TCP/IP.

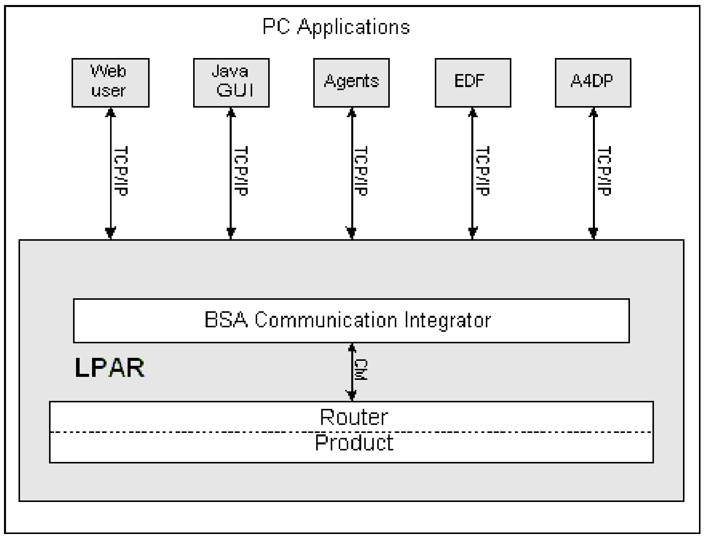
Multi-CPU capability

BSA CI can be installed on different CPUs. For details see the chapter on XCF (see "Multi-CPU using BSA XCF in a sysplex" on page 116) and on the Beta X-System Router (Beta 02) (see "BSA TCP/IP server as a standalone STC (Beta 02)" on page 144).

System overview

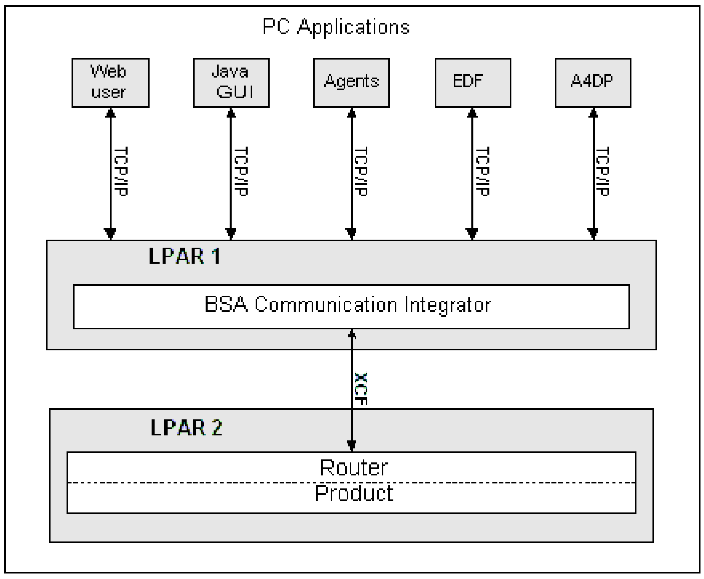
Single LPAR

BSA CI and the Beta Systems product can be installed on the same LPAR and communicate by means of BSA Cross Memory (CM):

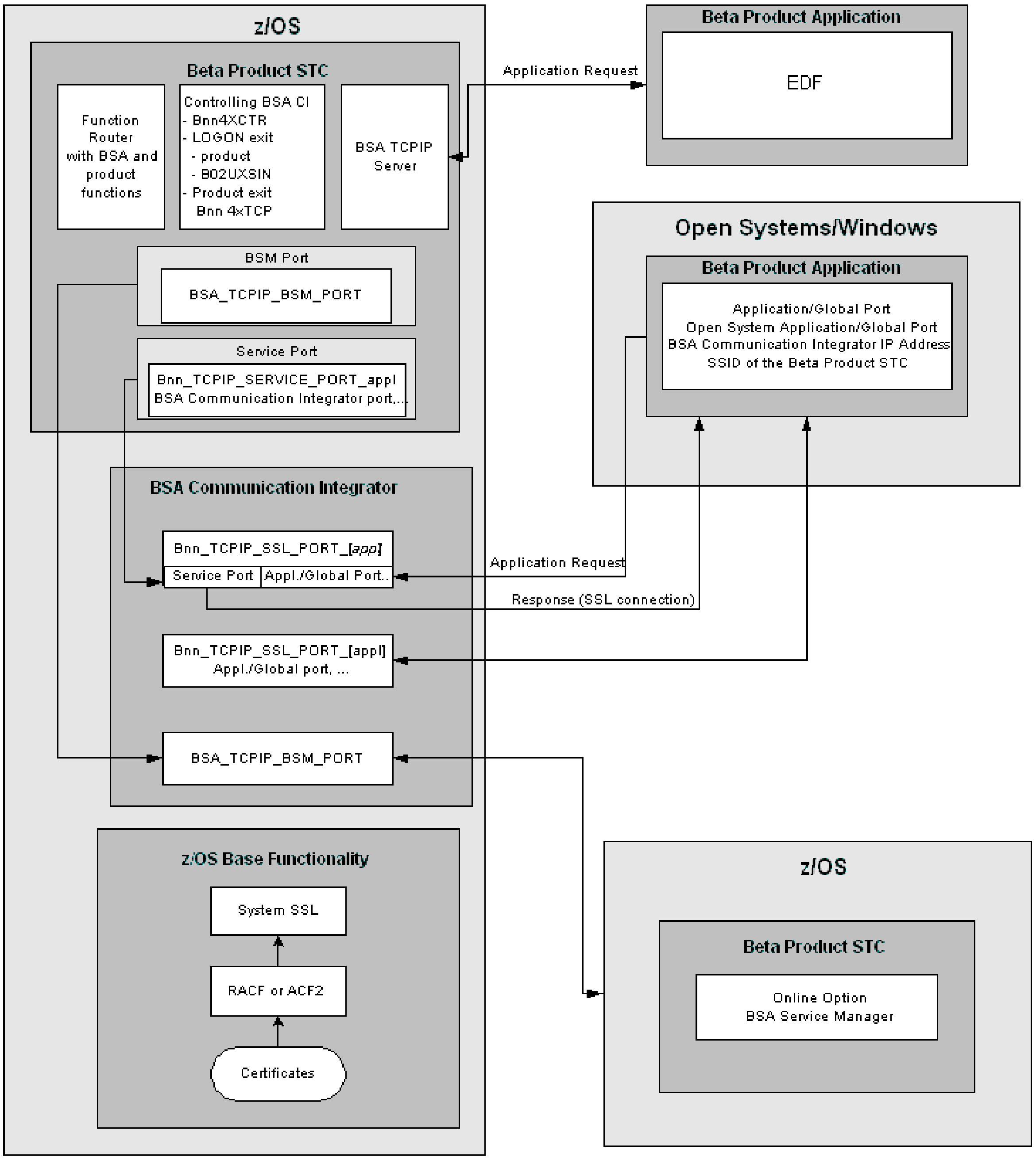


More than one LPAR

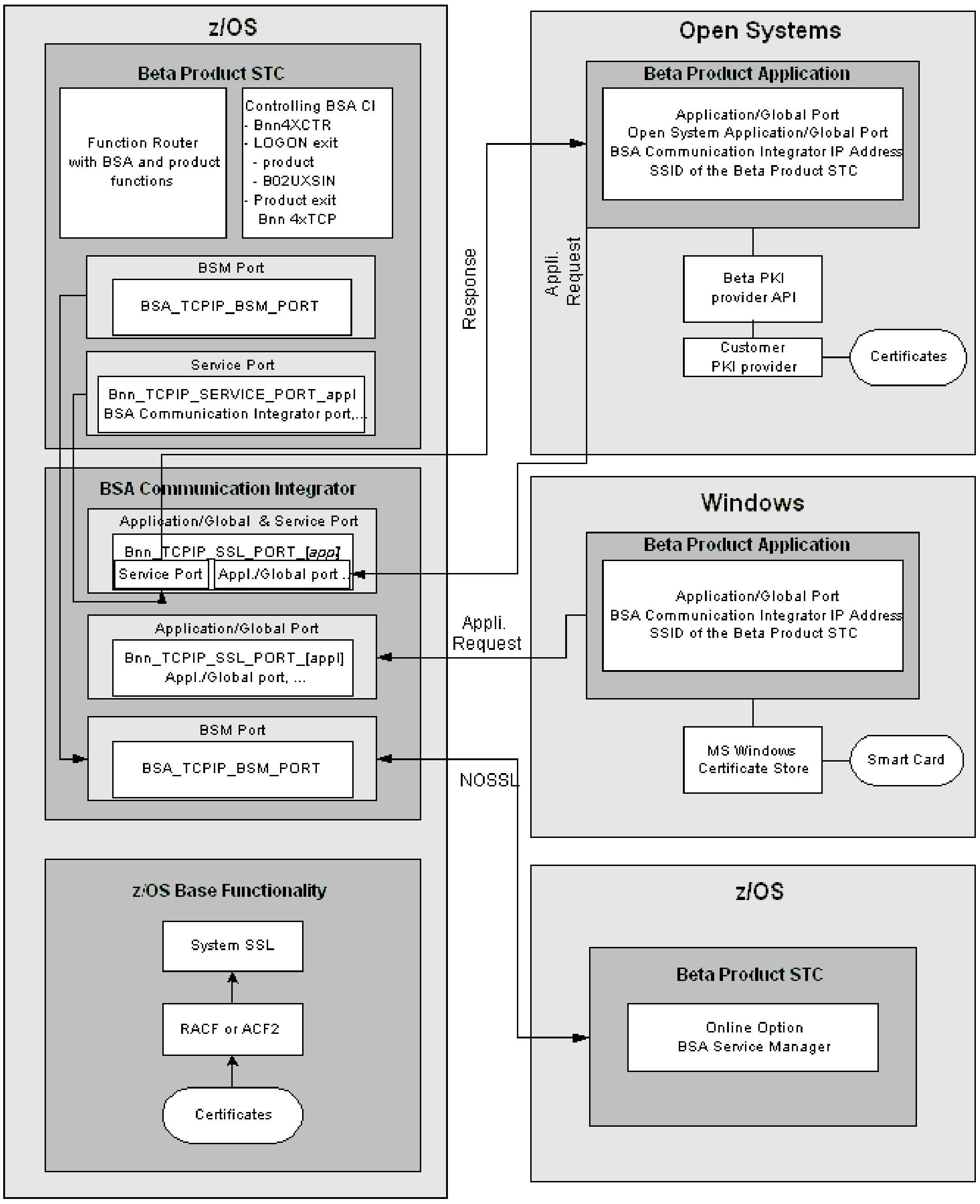
BSA CI and the Beta Systems product can be installed on different LPARs and communicate by means of BSA XCF:



NOSSL communication



SSL communication



BSA CI functions and features

NOSSL/SSL

BSA CI supports TCP/IP cross-platform communication between various systems, for example UNIX and z/OS, and allows online access from Beta Web applications. Transport can be unencrypted (NOSSL) or encrypted using IBM's System SSL (SSL). If you decide to use SSL, please make sure that the client program you are using also supports SSL.

Supported port types

BSA CI supports the following types of port:

- **global ports** (see page 185)
- **application ports** (see page 185)

  the port used by a product application
- **service ports** (see page 186)

  the port used for internal communication between the product server and the product application, e.g. to process specific service and administration requests.
- **BSM ports** (see page 187)

  These are special service ports used by the BSA Service Manager to provide ISPF online information on the operating status of BSA CI, and to enable to operating status to be modified.

Dynamic control

The TCP/IP functionality of BSA CI can be controlled by means of the BSA Service Manager and the appropriate LST parameters. The following are available:

- Display of port and user information
- Automatic retry is activated when, for example, the requested TCP/IP started task is not active in z/OS, or when the connection to the task has been interrupted due to an error.

DNS

Domain name system support for symbolic IP addresses with a maximal length of 255 characters.

Product exits

Activation of product-specific security exits that are defined and activated in the product started task. These exits are defined and executed in the started task, and are called via the BSA Service Manager.

Encryption/compression

In runtime mode NOSSL, encryption and/or compression are used for TCP/IP connections.

Runtime modes

Several runtime modes (see page 188) are supported for the relevant application and service ports.

User administration

In order to work with a product, the user needs to log onto the security system used by the target system. The logon is made in the runtime mode defined for the port concerned. Following successful logon, the user is administered by BSA CI.

Message routing

Special messages (in particular LOGON messages) are written to a special message table of the connected product, provided that database table MSG has been created for the product concerned.

Installing and customizing BSA CI

Prerequisites

BSA CI requires an OMVS segment. You can use a separate OMVS segment, or you can use the default OMVS segment.

Prerequisites for using SSL

The following prerequisites are only of interest if you want to use runtime mode SSL.

- IBM System SSL
- Public and private keys for the SSL connection

  SSL provides data privacy and integrity as well as client and server authentication based on public-key certificates. For each SSL connection, SSL uses a public and a private key.
- Key rings

  There is a PKI mechanism for authenticating each side of the connection and for agreeing on encryption keys. These keys are generated and stored in key databases, known as key rings.
- X.509 certificates

  X.509 certificates, containing public keys, are also required. The X.509 certificates can be created or requested and obtained. In either case, a certificate is then subsequently linked to or associated with and becomes part of a key ring.

SSL client certificate support

If you plan to implement SSL client certificate support, you must also have CA certificates from each certifying authority that verifies your client certificates. The certificate authority (CA) belongs to a started task, i.e. the STC is the owner of a CA. Several CAs can be used within BSA CI at the same time. A CA is the issuer of certificates and keys. RACF and ACF/2 can also be used as a CA and issue certificates. For each CA, a member for the key ring and one for the port definition must be available.

You must create a CA, make it HIGHTRUST, and add a key ring belonging to BSA CI to the CA. Next, create a certificate for BSA CI and connect it to the key ring.

For detailed instructions on the SSL options mentioned in the following procedure, see "Defining SSL authentication security" on page 192.

Procedure

Do the following to install BSA CI (You can ignore steps 1 and 3 if you are using NOSSL):

1. **For SSL only**: Define a security environment and the certificates (see page 193).
2. Create JCL for the BSA CI started task (see page 184).
3. **For SSL only**: Create and customize a member for the key ring and for the key-related parameters (see page 200).

   Note: Create a member for each key ring in use.
4. Create and customize a member for the port and for other parameters (see page 203).
5. Optional: Customize the logon procedure and the product exit.
6. Define a user ID (COMUID) and associate it with the BSA CI started task.

   Note: The user ID of the STC requires an OMVS segment (separate OMVS segment or default OMVS segment). READ access to the facility class BPX.DEFAULT.USER is required.
7. Define default user for RACF.

   To set up RACF so that it automatically uses default OMVS segments for users and groups that do not have their own OMVS segments in their USER or GROUP profiles, proceed as follows:

   1. Create a FACILITY class profile called BPX.DEFAULT.USER with universal access READ and APPLDATA(*defaultuser/ defaultgroup*).
   2. Define a RACF user profile containing an OMVS segment for *defaultuser*.
   3. Define a RACF group profile containing an OMVS segment for *defaultgroup*.

Creating JCL for the BSA CI started task

JCL for the BSA CI started task

```
+--------------------------------------------------------+
|//BSA#CI PROC                                           |
|//stepname EXEC PGM=BST02TSS,REGION=0M,MEMLIMIT=10G     |
|//*                                                     |
|//STEPLIB  DD  DISP=SHR,                                |
|//             DSN=BSA.LOAD                             |
|//LSTPARM  DD  DISP=SHR,                                |
|//             DSN=BETA.PARMLIB(lstparm)                |
|//BxxOSEC  DD  DISP=SHR,                                |
|//             DSN=BETA.PARMLIB(keyparm)                |
|//BSATRACE DD  SYSOUT=*                                 |
|//                                                      |
+--------------------------------------------------------+
```

DD statements

- DD STEPLIB defines the BSA load library.
- DD LSTPARM defines the startup parameter member.
- **Only for runtime mode SSL**: DD B*xx*OSEC defines the key ring definition file for SSL connections. The default DD statement name is BSSOSEC. You can freely choose the product DD statement name. Please see the description of LST parameter B*nn*_TCPIP_SSL_PORT_*app* (see page 204).

Sample JCL can be found in BSA.SAMPLIB(BSA#SSL).

Example

```
+--------------------------------------------------------+
|//BSA#SSL  PROC                                         |
|//BSA#SSL  EXEC PGM=BST02TSS,REGION=0M,MEMLIMIT=10G     |
|//STEPLIB  DD  DISP=SHR,DSN=BSA.LOAD                    |
|//BSSOSEC  DD  DISP=SHR,DSN=BSA.SAMPLIB(SSLKEY)         |
|//B92OSEC  DD  DISP=SHR,DSN=BSA.SAMPLIB(SSLKEY92)       |
|//LSTPARM  DD  DISP=SHR,DSN=BSA.SAMPLIB(SSLLST92)       |
|//CEESNAP  DD  SYSOUT=*                                 |
|//SYSUDUMP DD  SYSOUT=*                                 |
|//SYSPRINT DD  SYSOUT=*                                 |
+--------------------------------------------------------+
```

BSA CI ports

Overview

BSA Communication Integrator (BSA CI) supports global ports and application ports.

A global port can be used by different add-ons of the same product.

An application port can only be used by one add-on of a product.

You can use both global ports and application ports in the same configuration.

EDF applications

As a rule, EDF can only communicate with the BSA TCP/IP server.

BSA CI does not support EDF applications, unless this is specifically described in the product (for example Beta 92).

Application ports

Different applications (add-ons) may need to be controlled differently, therefore nearly all keywords can be given a suffix *app*. This enables a particular value to be used to control the add-on, as opposed to the default value specified by the keyword without the suffix *app*. The suffix *app* must have a length of 3 characters, and must be identical to the add-on shortname (e.g. EIF, AUD, EUI, …), and the 3-digit article name in the license file.

Global port

Keywords without a suffix always reference the global port, i.e. the global port is identified by the port statement without a suffix.

Client identification

When it connects, a client application (add-on) must identify itself. Otherwise the request that follows will be denied.

Encryption and compression

Encryption can be controlled via the LST parameters B*nn*_TCPIP_ENCRYPT and B*nn*_TCPIP_ENCRYPT_*app*. Compression can be controlled via the LST parameters B*nn*_TCPIP_COMPRESS and B*nn*_TCPIP_COMPRESS_*app*.

Use B*nn*_TCPIP_ENCRYPT and B*nn*_TCPIP_COMPRESS if you want to set values for all applications of one product. Use B*nn*_TCPIP_ENCRYPT_*app* and B*nn*_TCPIP_COMPRESS_*app* if you want to set values for an individual application of one product.

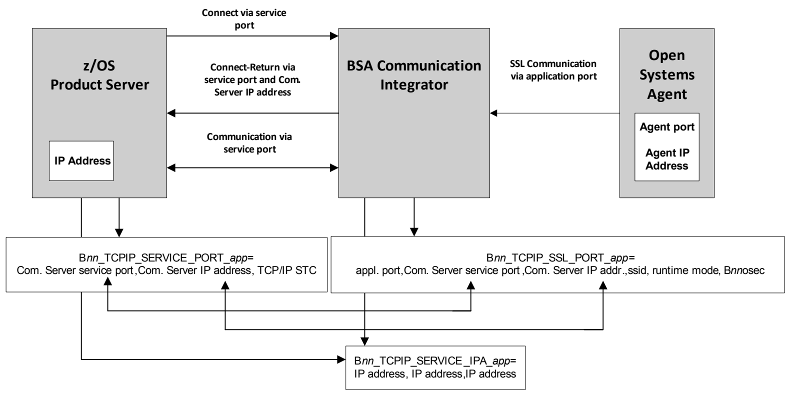
Defining a service port

Overview

The service port is used for the internal communication between the product server and the product application, for example, to process specific service and administration requests.

- The BSA Service Manager uses the service port to provide ISPF online information on the operating status of BSA CI, and to enable the operating status to be modified.
- The product server uses the service port for SSL communication with a product client agent to process specific administrative tasks.

Diagram


Connect via service port
z/OS Product Server
IP Address
Connect-Return via service port and Com. Server IP address
Communication via service port
BSA Communication Integrator
SSL Communication via application port
Open Systems Agent
Agent port
Agent IP Address
B*nn*_TCPIP_SERVICE_PORT_*app*= Com. Server service port ,Com. Server IP address, TCP/IP STC
B*nn*_TCPIP_SSL_PORT_*app*= appl. port,Com. Server service port ,Com. Server IP addr.,ssid, runtime mode, B*nn*osec
B*nn*_TCPIP_SERVICE_IPA_*app*= IP address, IP address,IP address

Assigning the service port

The service port is assigned to an application port of the BSA CI using the LST parameter B*nn*_TCPIP_SSL_PORT_*app* (see page 204).

Security

The port can be protected by a firewall, and it is possible to ensure that the port can only be used by authorized servers. To do so, use the LST parameter B*nn*_TCPIP_SERVICE_IPA[_*app*] (see page 204).

The inherent security is that the port only processes the requests that are sent by one or more previously defined IP addresses.

The IP address (ipa) stands for the relevant z/OS system where the z/OS product server concerned is active.

Dynamic server control and automatic retry

Overview

An operator command (z/OS MODIFY command) and various BSA Service Manager functions are available for controlling the server.

Operator command

The operator command **TL** can be used to view a list of all ports.

BSA Service Manager

Provided that a special service port has been defined (see next paragraph), the following BSA Service Manager functions are supported:

- Display information on users and ports.
- Cancel a user.

Port for BSA Service Manager (BSM port)

In order to enable access from a product via the BSA Service Manager, a special service port must be defined in BSA CI. This port is known as the "BSM port" and is defined in parameter BSA_TCPIP_BSM_PORT. This keyword must also be defined with exactly the same values in the product STC from which the Service Manager is to be called. It is also possible to use the Service Manager to dynamically insert the keyword into the product beforehand. Please refer to the *BSA Service Manager* documentation.

Automatic retry

Automatic retry is activated when, for example, the requested TCP/IP started task is not active in z/OS, or when the connection to the task has been interrupted due to an error. Retry parameters are set using keyword B*nn*_TCPIP_RETRY_INTERVAL (see page 204).

Available runtime modes

Overview

BSA CI supports the following runtime modes for ports:

- NOSSL
- SSL
- SSLAUTH

You can assign a different runtime mode to each port. The runtime mode of a port is specified as a positional parameter of the LST parameter B*nn*_TCPIP_SSL_PORT[_*app*] in the LST member of the BSA CI.

NOSSL

- NOSSL disables SSL in the BSA CI for this port.
- The user logon is carried out via user ID/password.
- No SSL handshake or certificates are exchanged.
- Encryption and compression can be activated.
- If the product application is a client, for example, a Beta 92 Open Systems (OSY) agent, communication is established without using SSL and/or certificates.
- Authentication is not checked before establishing communication. Certificates are neither requested nor checked. Client authorization is checked via user ID/password within the z/OS security system. All subsequent communication will be processed in accordance with the compression and/or encryption selected.

Note: The runtime mode of the service port is always SSL, even if you use NOSSL for the application port. For more information, see "Defining a service port" on page 186.

SSL

- This is SSL without client certificates.
- The server always sends its certificate to the client for the client to check (BSA CI always acts as the server during an SSL handshake).
- The user logon is carried out via user ID/password.
- When the product application is a client, e.g. a Beta 92 Open Systems (OSY) agent, certificate verification is sufficient to establish a communication.
- For communication to take place, only a server certificate must be made available. Client certificates, if available, are neither requested nor checked.
- When communication has been successfully established, all further communication is SSL-encrypted and the client authorization is carried out via user ID/password check within the z/OS security system.
- Runtime mode SSL is defined for the application port and the service port.
- The client must support SSL, and SSL support must be set as the runtime mode on the client side.

SSLAUTH

- This is SSL with server authentication and client authentication.
- The server and the client mutually exchange and check certificates.
- The user logon is carried out via a client certificate and no longer via user ID/password.
- When the product application is a client, e.g. a Beta 92 Open Systems (OSY) agent, the exchange of certificates is sufficient to establish a communication.
- Valid client and server certificates are required and must be made available before communication can take place. Communication is only established if the client authentication is valid. For client authorization (logon to the z/OS security system, e.g. RACF), the client certificate is used. HostIdMapping can also be used to map a client certificate to a z/OS user ID. If the client certificate is invalid or if it cannot be mapped to a valid z/OS user ID, communication cannot be successfully established.
- When authorization and authentication of the connection is successful, all further requests are processed SSL-secured.
- The client must support SSL, and SSL support must be set as the runtime mode on the client side.

Note: The runtime mode of the service port is always SSL, even if you use SSLAUTH for the application port. For more information, see "Defining a service port" on page 186.

Encryption and compression

- If you are using application ports with runtime mode NOSSL, you can specify the application suffix in the compression and encryption parameters, for example, B*nn*_TCPIP_COMPRESS_*app* (see "LST parameters for BSA CI" on page 204).
- Encryption and compression cannot be used for runtime modes SSLAUTH and SSL, and will be ignored.

Client authentication and authorization

Overview

The authentication and authorization modes used by the BSA CI depend on the selected runtime mode: NOSSL, SSL, or SSLAUTH.

Runtime modes SSL and NOSSL

In runtime modes SSL and NOSSL, authentication through BSA CI always takes place in the product STC by means of a RACF call. The installation of the logon security exit B02UXSIN determines how the RACF call works. For more information on B02UXSIN, see "BSA TCP/IP server logon exit (B02UXSIN)" on page 142.

The exit must be activated in the product STC. To do this, specify `UXSIN=B02UXSIN` in the B*nn*SSIxx LST member of the product STC. For more information on the UXSIN parameter, see "SSI parameters" on page 145.

Runtime mode SSLAUTH

In runtime mode SSLAUTH, there are two ways to check client authentication (logon to the Beta Systems product/z/OS) and authorization:

- Mapping and authentication (logon) of the user ID as defined in the z/OS security system on the basis of the allocation of certificate to user ID in the security system.
- Automatic mapping and authentication (logon) of the user ID defined in the certificate by means of certificate extension HostIdMapping (OID: 1 3 18 0 2 18 1) against the user ID defined in the z/OS security system. HostIdMapping is an IBM extension.

Defining SSL authentication security

Overview

To define SSL basic authentication security, you must first request or create a signed certificate for your BSA CI, and a certificate authority (CA) certificate from the issuer that signed your BSA CI certificate.

After you have received or created a signed certificate for your BSA CI and the associated CA certificate, you will need to the following:

- Authorize the use of digital certificates in RACF
- Store the BSA CI certificates and the BSA CI key rings

What to do on the client side

On the client side, you must create a key ring and attach to it the CA certificate from the certifying authority that issued the BSA CI's certificate. In the case of a z/OS client, you must use RACF to create a client key ring and then attach the CA certificate to the said key ring. For the client to authenticate the BSA CI, the BSA CI (more specifically, the controller user ID) must possess a signed certificate created by a certifying authority.

Server authentication

To prove its identity to the client, the BSA CI passes the signed certificate. The client must possess the CA certificate from the same certifying authority that issued the BSA CI's certificate. The client uses the CA certificate to verify that the BSA CI's certificate is authentic. Once verified, the client can be assured that messages come from that BSA CI, and not from anywhere else.

Client authentication

For the BSA CI to authenticate the client, note that there is no client certificate that the client forwards to prove its identity to the BSA CI. In the SSL basic authentication scheme (= runtime mode SSL), the BSA CI authenticates the client by demanding a user ID and password from the client.

Defining a security environment and certificates

Overview

The examples in this section show how to use the RACDCERT command to define the SSL security environment for the BSA CI, and how to specify that you are working with certificates.

RACDCERT

The RACDCERT command is the primary administrative tool for managing digital certificates using RACF.

The RACDCERT command is used to manage resources in the following classes:

DIGTCERT	Profiles in the class DIGTCERT contain information on digital certificates as well as the certificate itself. The user ID associated with the certificate can be found in the APPLDATA field of the profile.
DIGTRING	Profiles in the class DIGTRING contain information on key rings and on certificates that are part of each key ring. Key rings are named collections of the personal certificates, site certificates, and CA certificates associated with a specific user.
DIGTNMAP	Profiles in class DIGTNMAP contain information on certificate name filters.

Profiles in the classes DIGTCERT, DIGTRING and DIGTNMAP are automatically maintained through RACDCERT command processing. You cannot administer profiles in these classes using the RDEFINE, RALTER, and RDELETE commands.

RACF prerequisites

Make sure that the DIGTCERT and DIGTRING classes are active before defining certificates or key rings to RACF. If necessary, activate these classes:

```
SETROPTS CLASSACT(DIGTCERT DIGTRING)
```

Be sure to perform a refresh after each update or change:

```
SETROPTS RACLIST (DIGTRING DIGTCERT) REFRESH
```

Also ensure that the RACDCERT command is defined as an authorized TSO command in the IKJTSO*xx* member.

In order to issue the RACDCERT command, you must have access to the FACILITY class IRR.DIGTCERT function with UPDATE or CONTROL access.

Example of using RACDCERT command

```
+-----------------------------------------------------------------+
|jobcard                                                          |
|//RACFDEF  EXEC PGM=IKJEFT1A,DYNAMNBR=nn                         |
|//SYSTSPRT DD  SYSOUT=*                                          |
|//SYSLBC   DD  DISP=SHR,DSN=SYS1.BRODCAST                        |
|//SYSUADS  DD  DISP=SHR,DSN=SYS1.UADS                            |
|//SYSPRINT DD  SYSOUT=*                                          |
|//SYSTSIN  DD  *                                                 |
|       RACDCERT....                                              |
|/*                                                               |
+-----------------------------------------------------------------+
```

Define a communication ID

Define a user ID (for example, COMUID) and associate it with the BSA CI started task.

The user ID must have its own OMVS segments or use the default OMVS segment. It will need READ access to the facility class BPX.DEFAULT.USER.

Defining default user for RACF

To set up RACF so that it automatically uses default OMVS segments for users and groups that do not have their own OMVS segments in their USER or GROUP profiles, proceed as follows:

1. Create a FACILITY class profile called BPX.DEFAULT.USER with universal access READ and APPLDATA(*defaultuser*/*defaultgroup*).
2. Define a RACF user profile containing an OMVS segment for *defaultuser*.
3. Define a RACF group profile containing an OMVS segment for *defaultgroup*.

Creating the CA certificate

Create the certificate authority (CA) certificate.

The following is an example of the command used to create this most important certificate:

```
RACDCERT GENCERT CERTAUTH            +
         SUBJECTSDN(CN('BETA-CA-ZOS') +
         O('BETA') C('DE'))           *
         WITHLABEL('BSA-ZOS-CA')
```

where:

- GENCERT creates a digital certificate and potentially a public or private key pair.
- CERTAUTH relates to the certificate of a certificate authority (CA).
- SUBJECTSDN specifies the subject's distinguished name. In the example, the following values are used:

  ```
  CN (common name) : BETA-CA-ZOS
  O  (organization): BETA
  C  (country)     : DE
  ```

- WITHLABEL specifies the label assigned to this certificate. If specified, this must be unique to the user ID with which the certificate is associated.

Note: The label name is stripped of leading and trailing blanks. If a single quotation mark is intended to be part of the label name, you must use two single quotation marks together for each single quotation mark within the string, and the entire string must then be enclosed within single quotation marks.

Make the CA certificate HIGHTRUST

The sub-keywords TRUST | NOTRUST | HIGHTRUST indicate whether the status of the certificate is trusted, not trusted, or highly trusted. Whether the certificate is not trusted or trusted depends on whether or not the certificate is valid and whether the private key has been compromised or not. Since highly trusted certificates are by definition trusted certificates, any certificate usage that was enabled by marking the certificate trusted will also be enabled by marking the certificate highly trusted. However, only CA certificates can be highly trusted. The keyword ALTER is used to change the status to HIGHTRUST, for example:

```
RACDCERT CERTAUTH ALTER(LABEL('BSA-ZOS-CA')) HIGHTRUST
```

Note: The value of the LABEL keyword is case-sensitive.

Create the key ring

The following is an example of how to use the RACDCERT command to create a key ring for the started task user ID running the BSA CI (in this example: COMUID):

```
RACDCERT ADDRING(BETA-RING) ID(COMUID)
```

where:

ADDRING creates a key ring. Only users can have a key ring. Key ring names become the names of RACF profiles in the DIGTRING class, and can contain only characters allowed in RACF profile names. Although asterisks are allowed in ring names, a single asterisk is not permitted.

Note: For a CA certificate, the user who owns the ring is the BSA CI. Lowercase characters are permitted. A key ring name can be up to 237 characters in length. Since only user IDs can have key rings, neither CERTAUTH nor SITE can be specified with ADDRING.

Connect certificate to key ring

The following is an example of how to use RACDCERT to connect the certificate to the key ring:

```
RACDCERT ID(COMUID)                 +
         CONNECT(CERTAUTH           +
                 LABEL('BSA-ZOS-CA') +
                 RING(BETA-RING)     +
                 USAGE(PERSONAL)     +
                 DEFAULT)
```

where:

- ID(userid) indicates that the certificate being added to the key ring is a user certificate and user ID is the user ID associated with this certificate. If the ID keyword is not specified, it defaults to the value specified or the default value on the RACDCERT command.
- CONNECT specifies that a digital certificate is being added to a key ring.
- CERTAUTH indicates that the certificate being added to the key ring is a CA certificate.
- USAGE allows the altering of the trust policy within the confines of a specific key ring. For example, a CERTAUTH certificate connected with USAGE(PERSONAL) can be used to demote a certificate-authority certificate so as to ensure that it is not used as a certificate authority in this key ring. It can be used as a user certificate if a private key is present. Typically, however, one is not present. Consequently, connecting a CERTAUTH certificate as USAGE(PERSONAL) is a way to mark it as NOTRUST for only this key ring. Also, a user certificate connected with USAGE(CERTAUTH) can be used to promote an ordinary user certificate to a CA certificate. It can then be used to authenticate user certificates for only this key ring.
- DEFAULT specifies that the certificate is the default certificate for the ring. Only one certificate within the key ring can be the default certificate. If a default certificate already exists, its DEFAULT status is removed, and the new certificate in question becomes the default certificate. If you want the specified certificate to be the default, DEFAULT must be explicitly indicated.

Other certificates

Once you have created a CA certificate, you can start issuing other types of certificates, for example, an SSL certificate for the BSA CI started task user ID (in this example: COMUID) needed to process handshakes with future client certificates belonging to external or internal users.

Create the BSA CI certificate

Use RACDCERT as in the following example to create the BSA CI certificate:

```
RACDCERT GENCERT ID(COMUID)                      +
         SIGNWITH(CERTAUTH LABEL('BSA-ZOS-CA')) +
         SUBJECTSDN(CN('betasystem.com')        +
                    O('BETA') C('DE'))          +
         WITHLABEL('BSA_ZOS_COMSVR')
```

where:

- SIGNWITH(CERTAUTH(LABEL('*labelname*')) specifies the certificate with a private key signing the certificate. The example above shows our CA certificate. If SIGNWITH is not specified, the default is to sign the certificate with the private key of the certificate that is being generated.
- COMSVR stands for the BSA CI.

Connect COMSVR to key ring

Connect the BSA CI certificate to the key ring. For example:

```
RACDCERT ID(COMUID) CONNECT(ID(COMUID)            +
                            LABEL('BSA_ZOS_COMSVR') +
                            RING(BETA-RING))
```

Note: We did not specify USAGE because the default value of USAGE is the same as in the certificate added, i.e. we have kept USAGE(CERTAUTH). We also omitted DEFAULT as we chose not to make the CA certificate the DEFAULT in this key ring.

Enable COMUID to act as CA

The following commands allow the BSA CI user ID to act as a CA, for example:

```
PE IRR.DIGTCERT.GENCERT CL(FACILITY) ID(COMUID) +
   ACC(CONTROL)
PE IRR.DIGTCERT.LIST CL(FACILITY) ID(COMUID) ACC(R)
PE IRR.DIGTCERT.LISTRING CL(FACILITY) ID(COMUID) +
   ACC(R)
```

Refresh the RACLIST class

Perform a refresh for the RACLIST class, For example:

```
SETROPTS GENERIC (FACILITY) REFRESH
SETROPTS RACLIST (FACILITY) REFRESH
SETROPTS GENERIC(DIGTRING DIGTCERT) REFRESH
SETROPTS RACLIST (DIGTRING DIGTCERT) REFRESH
```

HostIdMapping certificates

You can add a HostIdMapping extension to certificates you create for certain users, thus enabling you to specify the user IDs for logging onto particular servers (or hosts).

The HostIDMapping extension (OID 1 3 18 0 2 18 1) is an IBM extension that is also available for public use. RACF automatically maps a valid certificate to the RACF user ID provided in the extension.

Controlling an identity used for logon purposes is an essential security objective. Therefore, you must enable the BSA CI to accept logons from clients who have been issued certificates with HostIdMapping extensions by creating profiles in the RACF class SERVAUTH.

Define profile in class SERVAUTH

Define a profile in class SERVAUTH for each server (host) name you want your BSA CI to honor when accepting logons for certificates containing HostIdMapping extensions. The profile has the format IRR.HOST.*hostname*, where the value of *hostname* is usually a domain name, such as betasystems.com. This domain name must be entered in the entry for HostIdMap in the certificate extension but without the subject ID portion in the APPL section, for example:

```
RDEF SERVAUTH IRR.HOST.BETASYSTEMS.COM UACC(NONE)
```

Note: Ensure that your CA certificate is defined as HIGHTRUST. See "Make the CA certificate HIGHTRUST" on page 195.

Give READ to STC user

Give READ rights to the BSA CI started task user ID (in this example: COMUID):

```
PE IRR.HOST.BETASYSTEMS.COM CL(SERVAUTH) ID(COMUID) +
   ACC(R)
```

Activate and raclist class SERVAUTH

Now activate and raclist class SERVAUTH, for example:

```
SETR CLASSACT(SERVAUTH)
SETR RACLIST(SERVAUTH)
SETR RACLIST(SETRAUTH) REFR
```

Note: On a z/OS system, HostIdMapping is not honored if the target user ID is created **after** the start of the validity period for the certificate containing the HostIdMappings extension. Therefore, if you are creating user IDs specifically for certificates with HostIdMappings extensions, make sure that you create the user IDs before the certificate requests are submitted.

Creating and customizing a member for key ring and other parameters

Key ring/key label in DD BSSOSEC

To be able to use SSL for communication, the BSA CI in question needs at least one server certificate for authentication. The server certificate and other relevant certificates (like client certificates) are stored in a key ring file. The key ring file is generally located in the z/OS security system, for example, RACF. A label assigned to the server certificate is also stored in the security system.

The name and location of the key ring and other parameters necessary for controlling the BSA CI are defined in DD BSSOSEC by default. You can define a different DD name in the LST parameter B*nn*_TCPIP_SSL_PORT_[*app*]. If you work with multiple key rings, code multiple DD statement in the JCL of the BSA CI and assign each DD name to the corresponding port via B*nn*_TCPIP_SSL_PORT_[*app*] (see "LST parameters for BSA CI" on page 204).

Example

An example of how to use the parameters can be found in member SSLKEY00 in the BSA.SAMPLIB:

```
KEYRING = BETA-RING
KEYLABEL = "BSA_ZOS_COMSVR"
SHOWCERT = YES
SHOWPARAM = YES
```

Parameters in DD BSSOSEC

The following parameters are defined in DD BSSOSEC:

- **KEYRING**

  Name of the key ring containing the server certificate to be used.

  Format: [USERID/]name - where USERID represents the owner of the ring.

  Note: If the key ring name includes a forward slash (for example, "PRODCOMS/custring"), always enter the user ID before the key ring name to avoid definition errors.

- **KEYLABEL**

  Name of the key label of the server certificate to be used.

- **SHOWCERT**

  The local (server) and partner (client) certificate are output in the system log of the BSA CI.

  Format: [YES | NO], default: NO

- **SHOWPARAM**

  Output of the parameters of the DD statement assigned to a respective port.

  Format: [YES | NO], default: NO

- **LDAP_SERVER**

  specifies an LDAP server host name. Each host name can contain an optional port number separated from the host name by a colon (max. 1023 characters). The LDAP server is used to obtain CA certificates when validating a certificate and when the local database does not contain the required certificate. The local database must contain the required certificates if no LDAP server is specified.

  Even when an LDAP server is used, root CA certificates must be located in the local database since the LDAP server is not a trusted data source. The LDAP server is also used to obtain certificate revocation lists (CRL).

- **LDAP_USER**

  indicates the distinguished name to be used when connecting to the LDAP server.

- **LDAP_USER_PW**

  specifies the password to use when connecting to the LDAP server.

The following parameters define which protocols are accepted by the BSA CI. To reduce the risk of vulnerability (POODLE attack), fallback to SSL3 is disabled by default. TLSV11 and TLSV12 are available as of z/OS V2.1, or via PTF for z/OS V1.13.

- **SUPPORT_SSLV3**

  Format: [YES | NO], default: NO
- **SUPPORT_TLSV1**

  Format: [YES | NO], default: YES
- **SUPPORT_TLSV11**

  Format: [YES | NO], default: NO
- **SUPPORT_TLSV12**

  Format: [YES | NO], default: NO

Creating and customizing a member for port and other parameters

Port definitions and other parameters

The BSA CI supports ports for different add-ons (product applications) and different products (B*nn*).

- The definitions are made under DD statement LSTPARM.
- Parameters must have the prefix B*nn* (where *nn* stands for a product number) or BSA, and they can have a suffix _*app* (see "LST parameters for BSA CI" on page 204).
- When parameters are coded over more than one line (80 characters), you can use as many continuation lines as you like. Blanks will be ignored.

Example

```
B92_TCPIP_SSL_PORT_EUI=(5712,,10.10.60.100,,SSLAUTH,CUSTSEC)
```

Further examples on how to use the LST parameters can be found in member SSLLST00 in the BSA.SAMPLIB.

LST parameters for BSA CI

Overview

This section describes the LST parameters that are available for controlling the BSA CI.

***_app* suffix**

Some applications (add-ons) may need to be controlled differently from other applications. This is why you can define global ports, which can be shared by different applications of one product, and application ports, which are can only be used by the specified application of the product.

Keywords with the ***_app*** suffix define values for application ports, where ***app*** is the 3-character identifier of the application (add-on). Keywords without the ***_app*** suffix are generally valid for all applications of the product or define values for global ports. Most keywords can be coded with and without this suffix. Keywords with the prefix **BSA_** are valid for all applications and products.

Examples:

```
B93_TCPIP_SSL_PORT
B93_TCPIP_SSL_PORT_BWE
B92_TCPIP_SSL_PORT_OSY
B92_TCPIP_SSL_PORT_BWE
BSA_TCPIP_MAXTHREADS
```

SSL port

Keyword: B*nn*_TCPIP_SSL_PORT

Keyword: B*nn*_TCPIP_SSL_PORT_*app*

Value	Description	Option	Default
(*port*,[*sport*], *ipa*,*ssid*, *mode*[,*ddn*])	Values are coded as a comma-separated list of positional parameters. The port number, IP address, product subsystem ID, and runtime mode are required. Other parameters are optional. Each parameter is described in more detail below:		
port	The port used by the product application to communicate with the Beta Systems product.	required	
sport	The service port that enables the z/OS product to send specific service requests to a product client agent. The entry is only necessary if the product uses such clients (also see LST parameter Bnn_TCPIP_SERVICE_IPA). If this port is used, BSA CI must be customized for SSL support.	optional	
ipa	The IP address (a symbolic name of up to 255 characters) of the BSA Communication Integrator.	required	
ssid	Reserved for future use.	n/a	
mode	The runtime mode of the application port. Possible values are SSLAUTH, SSL, NOSSL.	required	
ddn	The name of the DD statement used to define the key ring and key label definitions for the application and service ports.	optional	BSSOSEC

Authorized *sport* addresses

Keyword: B*nn*_TCPIP_SERVICE_IPA

Keyword: B*nn*_TCPIP_SERVICE_IPA_*app*

Value	Description	Option	Default
(*ipa*[,*ipa*][,*ipa*]...)	Defines all the IP addresses that are authorized to use the service port *sport* Use this keyword to ensure that the service port can only be used by authorized servers. In addition, the service port can be secured by a firewall.	optional	

Logon exit

Keyword: B*nn*_TCPIP_LOGON_EXIT

Keyword: B*nn*_TCPIP_LOGON_EXIT_*app*

Value	Description	Option	Default
name	Name (max. 8 digits) of a logon exit that is called by logon exit B02UXSIN The default logon behavior of the clients can be changed in the exit.	optional	

Max. number of threads

Keyword: BSA_TCPIP_MAXTHREADS

Keyword: B*nn*_TCPIP_MAXTHREADS_*app*

Before specifying this parameter, it is important to first ascertain the following:

- How many can threads the system can handle?
- How many threads must be available to avoid having the communication system come to a standstill?

Value	Description	Option	Default
50..99999	BSA_TCPIP_MAXTHREADS specifies the maximum number of threads to be handled globally by the Communication Integrator at any one time. • Parameter information messages are issued at startup. • Once the max. no. of threads is reached, the server waits 5 secs before attempting to open the next thread. A message is issued every 3 minutes.	required	200
2 - maximum specified for global limit	B*nn*_TCPIP_MAXTHREADS_*app* specifies the maximum number of threads to be handled by the Communication Integrator specifically for each application, for example OSY. • The number of threads specified for applications must not exceed the global maximum. • Parameter information messages are issued at startup. • Once the max. no. of threads is reached, the server waits 5 secs before attempting to open the next thread. A message is issued every 3 minutes. **Note:** If the BSA global port is used, this limit will always apply, no matter which application logs on via this port.	optional	200 or the maximum specified for global limit

Timeout **Keyword**: B*nn*_TCPIP_SESS_TIME_LIMIT

Value	Description	Option	Default
nnnnnnnnn	The maximum period of time (time limit) in minutes a user can remain inactive before being timed out. When the time is exceeded, the user will be disconnected and must relog onto the z/OS system before the user can proceed. The value defined via this keyword also applies to all application ports defined for the product (i.e. ports with suffix ***_app***).	optional	60

Retry **Keyword**: B*nn*_TCPIP_RETRY_INTERVAL

Value	Description	Option	Default
ttttttttt, zzzzzzzzz	The time and the number of the retry intervals (max. 9 digits each). If the task is not active or the connection is lost, an attempt will be made to automatically reconnect the server to the TCP/IP started task after the time specified by t has elapsed. This will be repeated z number of times. **t** Time in seconds the system is to wait before a reconnect is attempted. **z** Number of retry intervals This number specifies how often a reconnect between a server and the z/OS TCP/IP stack or STC is attempted. If no reconnect to the system can be made within the specified number of retry intervals, the server will be terminated. The value defined via this keyword also applies to all application ports defined for the product (i.e. ports with suffix ***_app***). Specify **0** to prevent retries.	optional	60,0

Compression

Keyword: B*nn*_TCPIP_COMPRESS

Keyword: B*nn*_TCPIP_COMPRESS_*app*

Value	Description	Option	Default
method	This keyword is only used when runtime mode NOSSL has been defined for the associated application port. *method* describes the compression method to be used. This keyword excludes the use of keyword B*nn*_TCPIP_ CRYPT_EXIT. Only use this keyword if the product concerned uses compression. The following values are allowed: STD Beta-specific standard compression NO No compression required	optional	NO

For BSA_WLM_SUBSYS_TYPE, see "LST parameters for ZIF" on page 213.

Encryption

Keyword: B*nn*_TCPIP_ENCRYPT

Keyword: B*nn*_TCPIP_ENCRYPT_*app*

Value	Description	Option	Default
(*method* [,*key*])	The keyword is only used for the relevant application port when the runtime mode NOSSL has been defined. It specifies the encryption method to be used and the key to be applied. If no key is specified, a key based on the indicated method will be generated automatically during each connect. Keys must consist of HEX values only (0123456789ABCDEF) Use this keyword only if the product concerned uses encryption. The following values are allowed: DES DES algorithm; key length = 16 TRD Triple DES algorithm, key length = 48 AES Advanced encryption standard, key length = 64 (encryption mode CBC; fallback to Triple DES if client does not support AES) BFS Blowfish algorithm, key length >=16 and <=128, divisible by 2 NO No encryption required	optional	NO

SVC **Keyword**: BSA_SVC_NUMBER

Value	**Description**	**Option**	**Default**
nnn	Number of the BETA SVC that is to be used.	required	none

BSA Service Manager support **Keyword**: BSA_TCPIP_BSM_PORT

Value	**Description**	**Option**	**Default**
bsmport	Port used for communication with the BSA Service Manager of one or more products. This parameter must be specified if BSA Service Manager commands are to be executed in a different address space to the one they were initiated in.	(required)	none

Buffer data trace **Keyword**: BSA_TCPIP_TRACE_BUF

Value	**Description**	**Option**	**Default**
YES | NO	This keyword logs all buffer data from SEND or RECEIVE requests in dump format. Set this parameter to YES only if requested by Beta Systems support. **Note**: An enormous amount of data is produced. If you want to write these data to a SYSOUT class, use DD statement BSATRACE. If you do not use BSATRACE, all the data will be written to the system SYSLOG.	optional	NO

Compression and encryption for the BSA CI

Overview

The BSA compression and encryption feature for TCP/IP is available on z/OS and UNIX platforms. Designed exclusively for data exchange, it is based on the internal BSA communication interface, i.e. on special BSA control information that is put in front of the actual data. BSA encryption and compression are only available on ports without SSL.

Data exchange can take place:

- Between UNIX/NT platforms and z/OS
- Between the Web Enabler application server and z/OS

How it works

Compression and encryption are performed in:

- BSA Communication Integrator on z/OS
- BQL server on all BSA-supported UNIX platforms

During the procedure, the client and the corresponding server decide which protocol is to be used during the TCP/IP connect of the client. Thus the client adapts itself automatically to the supported platform. As a rule, the server decides how data are to be transmitted.

Controlling compression/ encryption

Compression and encryption can be controlled via the following LST parameters, which must be specified in the system where the TCP/IP server is running:

- B*nn*_TCPIP_COMPRESS[_*app*] = STD | NO

  Specify STD to use Beta standard compression.
  NO means no compression, which is the default.

- B*nn*_TCPIP_ENCRYPT[_*app*] = DES | TRD | AES | BFS | NO

  Specify the encryption algorithm:
  DES for the DES algorithm
  TRD for the triple DES algorithm
  AES for advanced encryption standard
  BFS for the Blowfish algorithm
  NO means no encryption, which is the default.

Code B*nn*_TCPIP_ENCRYPT and B*nn*_TCPIP_COMPRESS if you want to set values for all applications of one product. Code B*nn*_TCPIP_ ENCRYPT_*app* and B*nn*_TCPIP_COMPRESS_*app* if you want to set values for an individual application of one product.

zIIP Facility (ZIF)

In this chapter

Topic	Page
Introducing ZIF	212
LST parameters for ZIF	213

Introducing ZIF

Overview

ZIF (zIIP Facility) is a Beta Systems component that supports IBM's z9 Integrated Information Processor (zIIP), a special processor for IBM System z9 mainframes. The zIIP is used to offload and process certain selected functions from universal central processors. It is designed to optimize the use of available resources, and can thus contribute to a reduction in software costs.

BSA and Beta Systems products can use zIIP, for example, for offloading compression. Beta 92 can also use zIIP for the batch find function.

Prerequisites and restrictions

In order to use ZIF, the following prerequisites and restrictions must be observed:

- A zIIP processor must be available (hardware).
- The Beta Systems product must support the use of zIIP (for example, by using compression).
- A connection between the Beta Systems product (requestor) and the Workload Manager (WLM) must be available.

Enabling ZIF

ZIF is enabled by coding the corresponding LST parameter(s) in the parmlib member that is used by the product or component.

For example, to enable ZIF in the BSA X-System Router (Beta 02), code the parameters B02_WLM_SUBSYS_TYPE and B02_WLM_SUBSYS_NAME in the active B02LST*xx* member (see "LST parameters for ZIF" on page 213).

ZIF activation

The z/OS Workload Manager (WLM) handles the functionality that enables programs to route to zIIP the parts of their workload that run over preemptable enclave SRBs. Provided that the appropriate LST parameters have been set, the following takes place when a started task or RFF batch job is started:

- A connection to the z/OS WLM is set up automatically
- The BSA WLM/zIIP environment is set up
- The appropriate enclave is created
- A startup message (*xxx*9181I) is written to SYSLOG and JESMSGLG (see *BSA Messages and Codes*)

Offload value

The offload value for ZIF is 100%; i.e. 100% of the SRB is processed by the zIIP. If the zIIP is already operating at full capacity, the SRB will be processed by a universal central processor.

LST parameters for ZIF

Overview This section lists all the LST parameters relevant to ZIF.

Subsystem type

Keyword Beta *nn*: B*nn*_WLM_SUBSYS_TYPE

Keyword BSA CI: BSA_WLM_SUBSYS_TYPE

Value	Description	Option	Default
type	Type of subsystem, for example, **STC** This parameter is used by the Workload Manager to classify an enclave (service class). When this parameter is specified, the BSA WLM/zIIP environment is activated and the enclave is set up. The function name is always BSA_ENCL. Please note that the service factors of the STC are influenced by the corresponding settings in WLM.	required	none

Subsystem name

Keyword Beta *nn*: B*nn*_WLM_SUBSYS_NAME

Keyword BSA CI: B*SA*_WLM_SUBSYS_NAME

Value	Description	Option	Default
name	Name of the subsystem	optional	BETA*nn*

Automatic restart management (ARM) support

In this chapter

Topic **Page**

Enabling ARM support....215
Registering started tasks with ARM....216
Activating ARM....217
ARM policy....218

Enabling ARM support

Overview

BSA supports automatic restart management (ARM).

Automatic restart management is a z/OS recovery function that can improve the availability of specific batch jobs or started tasks. When a job or task fails, automatic restart management can restart the job or task without operator intervention.

Note

At present, BSA does not support cross-system restart. The started task or job must be restarted on the same system it was running on at the time of the failure. Please take this into account when defining the ARM policy.

Enabling ARM support

To enable started tasks and/or batch jobs to participate in automatic restart management, code the following keyword in member B01LST*xx*:

```
ARM_SUPPORT = type
```

where *type* is one of the following:

type ...	**Enables automatic restart management for ...**
STC	started tasks
BATCH	batch jobs
ALL	started tasks and batch jobs

Registering started tasks with ARM

Registration options

When ARM support is enabled, the started task or batch job registers with the automatic restart manager during initialization using these options:

```
REQUEST=REGISTER
ELEMENT=BETA$sysclone$jobname
ELEMTYPE=SFF@BETA
TERMTYPE=ELEMTERM
```

where:

- *sysclone* is a 1- or 2-character shorthand notation for the name of the z/OS system. Please refer to the relevant IBM documentation for a complete description of the SYSCLONE static system symbol.
- *jobname* is the name of the started task or batch job.

Messages

When you register the started task or batch job with ARM, message *xxx*9170I is issued (where *xxx* is the product identifier). If registration fails, an additional message, *xxx*9171E, is issued.

Event exit

You can use the event exit to prepare a system for the restart of an element or to prevent the restart of an element.

By default, the module IEFBR14 is used. To use a different event exit, specify the name of the exit in member B01LST*xx* using the following keyword:

```
ARM_EVENT_EXIT =  name
```

A sample exit in source form can be found in member BST01AEX in the BSA.SAMPLIB.

Activating ARM

Overview

A system must be connected to an ARM couple dataset with an active automatic restart management policy.

Procedure

To activate ARM, do the following:

1. Allocate and format an ARM couple dataset.

   A sample job can be found in member ARMCOUPL in the BSA.SAMPLIB.

2. Activate the ARM couple dataset using one of the following methods:

   - At IPL, via an entry in member COUPLE*xx*
   - Dynamically, via the following operator console command:

     ```
     SETXCF COUPLE,TYPE=ARM,PCOUPLE=dsname
     ```

3. Define an ARM policy.

   A sample job can be found in member ARMPOLIC in the BSA.SAMPLIB.

4. Activate the ARM policy using the following operator console command:

   ```
   SETXCF START POLICY,TYPE=ARM,POLNAME=policyname
   ```

ARM policy

Note

TERMTYPE must be ELEMTERM. (Cross-system restart is not supported.)

Restart order definitions must ensure that the started task is restarted before the batch job is restarted (for example, by assigning LEVEL 4 to the started task and LEVEL 5 to batch jobs.

Sample ARM policy

A sample ARM policy can be found in member ARMPOLIC in the BSA.SAMPLIB.

```
+----------------------------------------------------------------------+
|jobcard                                                               |
|//ARMPOL   EXEC PGM=IXCMIAPU                                          |
|//STEPLIB  DD  DISP=SHR,DSN=SYS1.MIGLIB                               |
|//SYSPRINT DD  SYSOUT=*                                               |
|//SYSIN    DD  *                                                      |
| DATA TYPE(ARM) REPORT(YES)                                           |
| DEFINE POLICY NAME(BETAGRP) REPLACE(YES)                             |
|   RESTART_ORDER                                                      |
|     LEVEL(4)                                                         |
|       ELEMENT_NAME(BETA*)                                            |
|   RESTART_GROUP(BETAGRP)                                             |
|     TARGET_SYSTEM(*)                                                 |
|       ELEMENT(BETA$TA$BETA93P)                                       |
|       TERMTYPE(ELEMTERM)                                             |
|         RESTART_ATTEMPTS(3,300)                                      |
|         RESTART_METHOD(ELEMTERM,PERSIST)                             |
|         RESTART_METHOD(SYSTERM,STC,'S BETA93P')                      |
|//                                                                    |
+----------------------------------------------------------------------+
```

Sample job for creating couple dataset

A sample job for creating a couple dataset can be found in member ARMCOUPL in the BSA.SAMPLIB.

```
+----------------------------------------------------------------------+
|jobcard                                                               |
|//ARMCPL   EXEC PGM=IXCL1DSU                                          |
|//STEPLIB  DD  DISP=SHR,DSN=SYS1.MIGLIB                               |
|//SYSPRINT DD  SYSOUT=*                                               |
|//SYSIN    DD  *                                                      |
| DEFINEDS SYSPLEX(PLEX1)                                              |
|     DSN(PLEX.ARMCPL01) VOLSER(ARM001)                                |
|     CATALOG MAXSYSTEM(32)                                            |
|  DATA TYPE(ARM)                                                      |
|     ITEM NAME(POLICY)  NUMBER(5)                                     |
|     ITEM NAME(MAXELEM) NUMBER(25)                                    |
|     ITEM NAME(TOTELEM) NUMBER(20)                                    |
| DEFINEDS SYSPLEX(PLEX1)                                              |
|     DSN(PLEX.ARMCPL02) VOLSER(ARM002)                                |
|     CATALOG MAXSYSTEM(32)                                            |
|  DATA TYPE(ARM)                                                      |
|     ITEM NAME(POLICY)  NUMBER(7)                                     |
|     ITEM NAME(MAXELEM) NUMBER(35)                                    |
|     ITEM NAME(TOTELEM) NUMBER(30)                                    |
|//                                                                    |
+----------------------------------------------------------------------+
```

For more information on ARM, see the IBM publication *z/OS Setting Up a Sysplex*.

Databases and database batch utilities

In this chapter

Topic	Page
Introduction	220
Global LST parameters for BQL	221
Dynamic database extension	224
Merging concatenated VSAM datasets	228
Sharing databases with the BQL share option	232
Serialization	235
Activating IBM's Global Resource Serialization (GRS)	236
Synchronization file (SYNC)	237
BnnCLSYN: Cleaning the SYNC file	239
Notes on model spool file definitions	241
JCL for BSA database batch utilities	242
BST05CMD: Updating the database definition file	244
BST05UPF: Updating file information in the database definition file	246
Example: Activating database mirroring	248
Example: Deactivating database mirroring	252
BST05LAU: Locking and unlocking a BQL database	254
Loading data into a BQL database (BST05DBL)	255
Unloading data from a BQL database (BST05DBL)	257
Formatting a database component (BST05FOR)	258
Building or rebuilding keys for a table (BST05DBL)	260
Deleting the contents of a table, key, or file (BST05DBL)	261
BST05ANA: Database analysis and repair utility	262
BST08OCP: Copying/Checking/Repairing/Restoring archive files	270
Using concurrent copy to back up a running BQL database	272

Introduction

Overview

All Beta products use a BQL database. This is a relational database that can be used by one Beta product STC, or shared by a number of Beta product STCs.

BQL share option

The BQL share option enables you to set up master – slave systems where the slave systems have the same rights as the master. In this case the slave systems access the database directly. For more details on the BQL share option, see "Sharing databases with the BQL share option" on page 232.

Other master – slave configurations

In addition to BQL share, there are other options for setting up master-slave configurations:

- Two or more STCs on the same LPAR can share the same database. The slave systems access the database via the master system.
- OCF or XCF.

Global LST parameters

Global LST parameters for BQL are coded in the B*nn*LST*xx* member of the product. For a list of available parameters, see "Global LST parameters for BQL" on page 221.

Global LST parameters for BQL

Parameter name	Value	Description	Option	Default
BQL_GROUPS	1..9	To obtain a higher level of parallel processing in an STC, you can specify a maximum of nine BQL groups. Five groups is recommended. **Note**: This parameter has no effect on batch jobs. Batch jobs always work with one BQL group.	optional	1
BQL_FUNCTION	*nnn*	Specifies the number of functions that can work in parallel within a group.		999
BQL_MASTER_SSID	*ssid*	Specifies the master system. In a master-slave system on one LPAR, or in an OCF/XCF connection, this parameter must always be specified on the slave with the SSID of the master system. This does not apply if you are using a shared database (see "Sharing databases with the BQL share option" on page 232).	optional	None
BQL_SHARE_OPTION	NO | SPOOL | ALL	Enables you to activate or deactivate the BQL share option (see page 234).	optional	NO
BQL_OPEN_FILES	YES | NO	This parameter applies to _beta doc|z plus only: If NO is specified, the SPOOL, CACHE, and INDEX files will not be opened until they are needed for the first time, i.e. they will not be automatically allocated and opened when the started task is started. **Note**: The files concerned must be set to a specific status for this parameter to take effect. Please contact Beta Systems support for details.	optional	NO
BQL_INDEX_ CACHEBUFFER	*nnnnn*	You can specify the index files for _beta doc|z plus. *nnnnn* indicates the number of buffers (in 4K blocks) to be used. If the value specified here is > 30000, and A2G_SUPPORT=YES has not been set, the value will be reset to 30000.	optional	0

Parameter name	Value	Description	Option	Default
BQL_SPEEDMASTER	YES | NO	YES specifies that the BQL Speedmaster function is to be used for all DATA and KEY files. You can define the number of buffers for each file in the database definition file. **Important**: Do **not** change the value of this parameter to NO. Otherwise serious performance problems could occur. **Exception**: You have to set this parameter temporarily to NO following the occurrence of message 9502E or 9503W. For more information, see "Operator Response" of message 9502E and 9503W in *BSA Messages and Codes*.	optional	YES
BQL_SPOOLCHECK	YES | NO	YES The master STC checks all spool blocks at startup and internal control blocks are built. Unused reader blocks are freed. NO There is no spool check at startup, which reduces the start phase of the master STC considerably. Read-only spool files are always checked at startup, irrespective of the value of this parameter.	optional	NO
BQL_STATISTIC	YES | NO	The BQL statistics functionality can be switched on and off. When a started task or batch job is stopped, the values found for tables and commands will be written.	optional	NO
BQL_QRY_ MAXMEMORY	0..2000	Central storage (amount in MB) Use this keyword to specify how much central storage is to be used for database queries online (online option **D.Q**). Specify **0** to use DASD storage instead of central storage. **Note**: If both BQL_QRY_MAXMEMORY and BQL_QRY_FILESPACE are set to **0**, unlimited central storage is used.	optional	If VDF: 2 If TSO: 2000
BQL_QRY_FILESPACE	0..2000	DASD storage (amount in cyls) Use this keyword to specify how much file space is to be used on a temporary DASD for database queries online (online option **D.Q**). Specify **0** to use central storage instead of DASD storage.	optional	50

Parameter name	Value	Description	Option	Default
BQL_DICT_FLAG_ SEARCH	YES | NO	By default, option **D.2.3** (FIELDS) includes subfield information (so-called flag fields) in the displayed table. If you don't need this information, you can disable this function to reduce resource consumption if the number of flag fields used by the product is very high (for example _beta access). This parameter can be updated dynamically via the Service Manager.	optional	YES
BQL_TRACE	YES | NO	YES causes the logging of all BQL and BOF commands	optional	NO
BQL_USER	number of users	This keyword specifies the number of users allowed in the internal RACF security user table of BQL. **0** means that the internal security table will not be used. When you increase or decrease the number of users, refresh the table with the following MODIFY command for your change to take effect: `F stcname,REFRESH RCF[,U=userid]` For more information, see "Refreshing the RACF security user table" on page 85.	optional	500

Dynamic database extension

Overview

The data and key files of product databases can be extended dynamically without having to interrupt normal operations. This enhances the availability of the product and contributes to achieving zero downtime.

To enable dynamic extension, the database definition must specify secondary space or volume candidates or both. The database is extended automatically when used space reaches a critical level (high water mark minus 1 percent).

Multi-CPU

Dynamic database extension is also possible in multi-CPU configurations that use the BQL share option, and in master-slave configurations.

Maximum size: 4 GB or 28 GB

The following applies to BQL databases:

- The total maximum size of a standard VSAM dataset is 4 GB.
- To enable extension beyond the 4 GB border, a dataset must be set up as an extended VSAM dataset. This is defined via the SMS data class. The maximum size of an extended VSAM dataset is 28 GB.

Minimum size

The minimum size of a database file is 5 cyl. This minimum value applies to primary space and it applies to secondary space. The minimum size does not apply to the SYNC file, which can be smaller.

Making databases dynamic

To enable dynamic database extension for a new database, do the following when defining the database:

- Specify secondary space or add volumes or both (see page 227)

To enable dynamic database extension for an existing database, do one of the following:

- "Adding volumes" on page 226
- Recommended: Define a new database with secondary space and copy the old database into the new database (see page 226)

Important: An existing VSAM dataset cannot be extended if concatenated datasets (EXT01, EXT02 etc.) are present. Follow the instructions in "Merging concatenated VSAM datasets" on page 228 in order to merge concatenated VSAM datasets into one dynamic database.

Dynamic extension

A dynamic database is extended automatically when the used space reaches a critical level (high water mark minus 1 percent). It is also possible to trigger dynamic database extension manually via the ISPF application (line command **X** in the "Dataset Definition Selection" table (option D.1)).

The process of dynamic database extension is indicated by the following messages:

```
9510I FORMAT OF 'dataset' TYPE: type ACTIVATED
9511I FORMAT OF 'dataset' TYPE: type ENDED SUCCESSFULLY
```

The dynamic extension of databases is handled within the STC. The database is locked against change requests during formatting. Mirror databases are also extended if present.

Format extend error

The dynamic extension of a database will fail if there is not enough free space on any of the specified volumes.

If dynamic extension fails due to lack of space or other reasons, the database receives the status FEX (Format EXtend Error). The database remains fully functional and work can continue, but the dynamic extension function is disabled for this database while it has the status FEX.

If the error is caused by lack of space on available volumes, create extra space on these volumes or "Adding volumes" on page 226. Depending on the product, you can also free space in the existing database via the product's cleanup jobs. Afterwards reset the status of the database (line command **RX** in the "Dataset Definition Selection" table (option D.1)) in order to enable its dynamic extension. The FEX status of the database is also reset when the STC is restarted.

Monitoring growth (MAXSIZE)

It is possible to monitor the growth of dynamic databases by defining a warning threshold. This value is defined in the database definition file via the MAXSIZE(*cyl*) parameter (see "BST05UPF: Updating file information in the database definition file" on page 246).

If MAXSIZE is defined for a data or key file, the system will write an additional message after it has terminated the extension of a database file:

```
9549I nn% OF MAXSIZE - 'dataset' (T: totalblocks M: maxblocks)
9549W DEFINED MAXSIZE (maxblocks/nn%) FOR 'dataset' HAS BEEN
REACHED
```

An informational message is written if the total size of the database is less than MAXSIZE and a warning message is written if it is greater or equal. The warning threshold and the current allocation (in percent) is also displayed online in the "Display Data Set Information" panel (option D.1).

Reaching the MAXSIZE value does not have any impact on the database, which remains fully functional and continues processing.

High water mark

Dynamic databases are extended automatically before used space reaches the high water mark (high water mark minus 1 percent). The occurrence of the high water mark message **9550W HIGH WATER MARK REACHED *nn% dsname*** for a dynamic database indicates that dynamic extension has failed (**9512E FORMAT OF '*dataset*' TYPE: *type* FAILED (RC: *rc* T: *tb*/*hwm*%**)). The high water mark message will continue to be written while the used space increases until you solve the problem that caused the format extension error and reset the FEX status of the database.

Adding volumes

You can use IDCAMS to add new volumes at any time during active operation, for example:

```
ALTER   TEST.QADOC.DATA
ADDVOLUMES(VOL002 VOL003 VOL004)
```

When a dataset with secondary space is extended and a new volume comes into use, the primary space is allocated, but only the secondary space is formatted. This may result in a difference between High Used RBA and High Allocated RBA. The next time the database is extended, the difference between High Used and High Allocated is formatted.

You can use line command **S** under option **D.1** to display the High Used RBA and High Allocated RBA of a database. These panels also display how many volumes have been defined and how many unused volumes are still available. See the *BSA Service Manager Manual* for details.

Making an existing database dynamic

Following is a description of the recommended method for making a database dynamic. This method corresponds to the steps of an ENLARGE job as generated under option **4** of the "Service and Database Selection Menu". You can specify the desired values for primary and secondary space in the corresponding panels. The STC must be down during this operation.

Procedure:

1. Back up your existing database.
2. Create a new database using IDCAMS. Specify secondary space to make this database dynamic.
3. Copy the old database into the new database using IDCAMS REPRO.
4. Optional: Format added space using BST05FOR with one of the following parameters:

   ```
   FORMAT FILE(dsname)
   ```

   -OR-

   ```
   FORMAT FILE SNAME sname
   ```

   where *dsname* is the dataset name and *sname* is the short name defined in the database definition file. This step is optional because formatting can also be carried out automatically by the active started task.

Creating a new dynamic database

Specify secondary space for your databases when you run the installation REXX if you want to create dynamic databases. The installation REXX will tailor your installation member S#*nn*UDEF with values for primary and secondary space accordingly, for example:

```
SPACE(20,10)
```

These values will be used by BST05FOR when it allocates and formats the database:

```
FORMAT FILE DSNAME(dsname)
```

-OR-

```
FORMAT FILE SNAME sname
```

You can later add volumes using IDCAMS (see "Adding volumes" on page 226).

Alternatively, you can also specify the volumes when you allocate the database using IDCAMS and later format it using BST05FOR:

```
DEFINE  CLUSTER(                                    -
           NAME(TEST.QADOC.DATA)                    -
      CYLINDERS(20 10)                              -
   SHAREOPTIONS(3 3)                                -
            VOL(VOL001 VOL002 VOL003)               -
          OWNER(BETA05)                             -
         UNIQUE                                     -
     NONINDEXED                                     -
               )                                    -
           DATA(                                    -
           NAME(TEST.QADOC.DATA.DATA)               -
 CONTROLINTERVALSIZE(4096)                          -
          RECORDSIZE(4088 4088)                     -
               )
```

Merging concatenated VSAM datasets

Overview

In previous versions of BSA, you could concatenate up to sixteen physical VSAM datasets to be used as one logical database. This was defined in the database definition file via ADD EXTEND. Concatenated datasets were used, for example, to overcome the 4 GB size limit of BSA V3 databases.

Important: Deprecated feature

Using concatenated VSAM datasets is deprecated and current versions of BSA provide only limited support for them. It is possible to continue to work with existing concatenated VSAM datasets, but current versions of BSA don't provide support for further enlargement.

We strongly recommend that all users who are still working with concatenated VSAM datasets merge these datasets. Use extended VSAM datasets if the total size is greater than 4 GB. This is defined via the SMS data class (DSNTYPE = EXT). For BQL databases, the maximum size of an extended VSAM dataset is 28 GB. Define secondary space and additional volumes for your merged VSAM dataset to enable the dynamic extension of this database (see "Dynamic database extension" on page 224).

Who is affected

Use the database panels under option **D.1** and **D.2.4** to find out whether you are affected. The table displays a **Y** in column **X** if concatenated VSAM datasets are present. Otherwise this column is blank.

Procedure

The following steps are needed to merge concatenated VSAM datasets into one. The sample JCL is for a database with one extend (EXT01). All database access (STC(s) and batch jobs) must be stopped during this procedure.

Before you begin:

- Create or adjust the data class (DSNTYPE = EXT) if the merged dataset is greater than 4 GB or if you want to enable dynamic database extension beyond 4 GB.
- Look up the short name of the VSAM dataset under option **D.1**. You need to know the short name of the database, for example, JOBDATA or B93LIST, for the DEL EXTEND command used in step 3.

Following are the steps you need to carry out:

1. Copy the contents of the VSAM datasets into sequential output files.

```
+--------------------------------------------------------------+
|//DBREPRO EXEC PGM=IDCAMS                                     |
|//IN1      DD DISP=OLD,                                       |
|//            DSN=VSAM.BETA.DB                                |
|//IN2      DD DISP=SHR,                                       |
|//            DSN=VSAM.BETA.DB.EXT01                          |
|//OUT1     DD DISP=(,CATLG),SPACE=(CYL,(600,50),RLSE),        |
|//            DSN=TEMP.BETA.DB,                               |
|//            DCB=(RECFM=FB,LRECL=4088,BLKSIZE=0)             |
|//OUT2     DD DISP=(,CATLG),SPACE=(CYL,(100,30),RLSE),        |
|//            DSN=TEMP.BETA.DB.EXT01,                         |
|//            DCB=(RECFM=FB,LRECL=4088,BLKSIZE=0)             |
|//SYSPRINT DD  SYSOUT=*                                       |
|//SYSIN    DD  *                                              |
|REPRO IFILE(IN1) OFILE(OUT1)                                  |
|REPRO IFILE(IN2) OFILE(OUT2)                                  |
|/*                                                            |
+--------------------------------------------------------------+
```

2. Rename the old VSAM datasets as a backup.

```
+--------------------------------------------------------------+
|//DBREN    EXEC PGM=IDCAMS                                    |
|//*                                                           |
|//SYSPRINT DD  SYSOUT=*                                       |
|//SYSIN    DD  *                                              |
| ALTER VSAM.BETA.DB                                      -    |
|       NEWNAME(VSAM.BETA.DB.OLD)                              |
| ALTER VSAM.BETA.DB.DATA                                 -    |
|       NEWNAME(VSAM.BETA.DB.OLD.DATA)                         |
| ALTER VSAM.BETA.DB.EXT01                                -    |
|       NEWNAME(VSAM.BETA.DB.EXT01.OLD)                        |
| ALTER VSAM.BETA.DB.EXT01.DATA                           -    |
|       NEWNAME(VSAM.BETA.DB.EXT01.OLD.DATA)                   |
|/*                                                            |
+--------------------------------------------------------------+
```

3. Delete the extends from the database definition file.

   You must carry out this step once for each extend. If you have more than one extend, you must carry out this step as often as the number of extends present.

```
+--------------------------------------------------------------+
|//DBFORM   EXEC PGM=BST01RFF,REGION=0M,COND=(0,NE),            |
|//             PARM=('S=nn,B01LST=nn,BnnLST=nn',               |
|//            'PGM=BST05CMD,SIGNON=NO')                        |
|//*                                                            |
|//STEPLIB  DD  DISP=SHR,                                       |
|//             DSN=BETA.BSAV7.LOAD                             |
|//SFFPARM  DD  DISP=SHR,                                       |
|//             DSN=BETA.PARMLIB                                |
|//BnnDEF   DD  DISP=SHR,                                       |
|//             DSN=VSAM.BETA.DEF                               |
|//*                                                            |
|//BQLIN    DD *                                                |
| DEFINE UPDATE FILE sname DEL EXTEND                           |
|/*                                                             |
|//SFFFDUMP DD  SYSOUT=*                                        |
|//SYSUDUMP DD  SYSOUT=*                                        |
|//*                                                            |
|//BQLPRINT DD  SYSOUT=*                                        |
|//SYSPRINT DD  SYSOUT                                          |
+--------------------------------------------------------------+
```

4. Create a database of the required size (primary and secondary space). If appropriate, list the potential volumes for it.

```
+--------------------------------------------------------------+
|//DBDEF    EXEC PGM=IDCAMS                                     |
|//SYSPRINT DD SYSOUT=*                                         |
|//SYSIN    DD *                                                |
| DEFINE      CLUSTER(                                     -    |
|                NAME(VSAM.BETA.DB)                        -    |
|           CYLINDERS(cyl,cyl)                             -    |
|        SHAREOPTIONS(3 3)                                 -    |
|                 VOL(vol1 vol2 vol3 vol4)                 -    |
|               OWNER(Bnn)                                 -    |
|              UNIQUE                                      -    |
|               SPEED                                      -    |
|          NONINDEXED                                      -    |
|                    )                                     -    |
|                DATA(                                     -    |
|                NAME(VSAM.BETA.DB.DATA)                   -    |
| CONTROLINTERVALSIZE(4096)                                 -    |
|          RECORDSIZE(4088 4088)                           -    |
|                    )                                          |
|/*                                                             |
+--------------------------------------------------------------+
```

5. Copy the contents from the sequential files created in step 1 to the new VSAM dataset created in step 4.

```
+-------------------------------------------------------------+
|//DBREPRO  EXEC PGM=IDCAMS                                    |
|//IN1      DD DISP=OLD,                                       |
|//            DSN=TEMP.BETA.DB                                |
|//         DD DISP=SHR,                                       |
|//            DSN=TEMP.BETA.DB.EXT01                          |
|//OUT1     DD DISP=OLD,AMP=('AMORG,BUFND=181'),               |
|//            DSN=VSAM.BETA.DB                                |
|//SYSPRINT DD SYSOUT=*                                        |
|//SYSIN    DD *                                               |
| REPRO IFILE(IN1) OFILE(OUT1)                                 |
|/*                                                            |
+-------------------------------------------------------------+
```

Sharing databases with the BQL share option

Overview

The Open Communication Facility OCF (see "Multi-CPU with the Open Communication Facility (OCF)" on page 86) and/or the Cross Coupling Facility XCF (see "Multi-CPU using BSA XCF in a sysplex" on page 116) are recommended for making the BQL service available in a multi-CPU environment. One BQL system (BQL_MASTER) is in charge of serialization and caching. The BQL_SHARE_OPTION is set to NO (default).

Alternatively, the BQL service can be made available by setting BQL_SHARE_OPTION to ALL (or SPOOL) if OCF and XCF are not an option.

Requirements and recommendations

Our general recommendation is to work with BQL_SHARE_OPTION=NO, which is the default.

Database sharing with BQL_SHARE_OPTION=ALL requires GRS. BETABQL reserves must be converted into a global enqueue as described in "Activating IBM's Global Resource Serialization (GRS)" on page 236.

BQL_SHARE_OPTION=ALL can also reduce performance because there is less caching and an increased overhead for data synchronization.

If OCF and XCF are not an option and you therefore consider to make use of BQL_SHARE_OPTION=ALL, you should contact Beta Systems support first.

Activating database sharing

Database sharing is activated by the master STC when it is started with the following LST parameters:

- BQL_SHARE_OPTION = SPOOL | ALL
- BQL_MASTER_SSID = *ssid_of_the_master*

If master and slave use different LST members, code the same parameters in each member.

Important: All database activities (master and slave STCs and all batch jobs) must be stopped **before** changing the BQL share option.

Please also be aware of the following **before** you change the BQL share option:

- The SYNC file holds the information that a database is shared, including the subsystem ID of the master.
- Database sharing is activated by the master STC when it is started with the appropriate BQL share option. Only the master can change the BQL share option in the SYNC file.
- When you clear the SYNC file, for example, by using job B*nn*CLSYN, the database sharing information will also be deleted.
- BQL_SHARE_OPTION = ALL disables BQL Speedmaster, which means that performance may suffer.
- Migrations and new installations should not take place when database sharing is active.
- We recommend that service programs such as CLEANUP and ARCHIVE be used exclusively on the master system.

LST parameters for the BQL share option

Parameter name	Value	Description	Default
BQL_SHARE_OPTION	NO	Only the master system is allowed to access the database. All database requests (BQL **and** SPOOL) from a slave system are sent to the master system, which processes these requests. • Code B*nn*DEF DD DUMMY for the slave STC if it is running on an LPAR that has no access to the database (because the corresponding disks are not shared). • In a multi-CPU environment, an OCF or XCF connection is required to enable communication across LPAR boundaries.	NO
	SPOOL	Both master and slave are allowed to access the SPOOL files. All BQL database requests from the slave system are sent to the master system, which processes these requests. • DD B*nn*DEF in the JCL of the master STC **and** the slave STC must reference the database definition dataset. • In a multi-CPU environment, an OCF or XCF connection is required to enable communication across LPAR boundaries.	
	ALL	Each system (master **and** slave) handles its database requests (BQL **and** SPOOL) by accessing the database directly. • DD B*nn*DEF in the JCL of the master STC **and** the slave STC must reference the database definition dataset. • An OCF or XCF connection is not required here. **Note on _beta access**: Although an OCF or XCF connection is not strictly required, its absence restricts shared functionality, and _beta access does not recommend using BQL_SHARE_OPTION = ALL without OCF/XCF.	
BQL_MASTER_SSID	*ssid*	This keyword specifies the subsystem ID of the master STC.	None

Notes

Always check the Beta product *Installation and System Guide* to find out whether there is product-specific information.

Serialization

Overview

The master subsystem synchronizes resource requests to the Beta product subsystem. It locks a resource when a batch job or subsystem is given access to this resource and places all other requests for this resource in a waiting list. After the resource becomes free, the batch job or subsystem which is next in the queue is given control.

Resource locking can take place on three different levels:

- The whole database can be locked (master lock).
- A single file can be locked (file lock).
- Other resources like tables and pools can be locked.

All serialization takes place in the address space of the master subsystem. If a requesting subsystem or batch job abends abnormally, the obsolete requests are deleted from the waiting list by the master subsystem. If a subsystem or batch job abends while it has control over a resource, the resource is unlocked by the master subsystem.

Activating IBM's Global Resource Serialization (GRS)

Overview

IBM's Global Resource Serialization (GRS) can also be used to synchronize databases. It is particularly important if you are using shared databases.

Using IBM's GRS in conjunction with the reserve conversion resource definition has the advantage of significantly enhancing performance. This is because not all of the DASD unit will be blocked, but only the datasets actually used by BQL.

BQL uses the macro RESERVE in connection with the major name BETABQL.

Example of an GRSRNLxx* entry:

```
+-------------------------------------------------------------+
|/***********************************************************/|
|/* RESERVE CONVERSION RESOURCE NAME LIST – RNLDEF STATEMENTS  */|
|/*(THE DEFAULT RESERVE CONVERSION RESOURCE NAME LIST IS EMPTY)*/|
|/***********************************************************/|
|RNLDEF RNL(CON) TYPE(GENERIC)                                |
|QNAME(BETABQL)                                               |
+-------------------------------------------------------------+
```

*	–	xx stands for a suffix member

Further information

For more information on GRS, see the appropriate IBM manuals.

Synchronization file (SYNC)

Overview

Access to the product database is controlled with the help of the SYNC file. Only one subsystem (either started task or batch job) has direct access to the product database at a given time. If other subsystems access the product database, they do so indirectly, namely via the subsystem which has control over the database.

Master subsystem

The subsystem which has direct access to the product database is called the master. When the STC or batch job is started, an entry is made in the SYNC file of the product database. This entry, which includes the SSID of the STC or batch job, gives the subsystem exclusive control over the database. When the STC is stopped or the batch job has terminated, the entry in the SYNC file is removed.

Slave subsystems

An entry in the SYNC file prevents any other subsystem from gaining direct access to the database. If other subsystems access the product database while it is under control of a master subsystem, they do so indirectly via the master subsystem. Subsystems which have only indirect access to a product database are called slaves.

Batch jobs

Specify SIGNON=YES in the JCL of the batch job for indirect access to the product database (slave).

Specify SIGNON=NO in the JCL of the batch job for direct access to the product database (master).

BQL_MASTER_SSID

The subsystem ID of the master subsystem must be specified in the LST parameter BQL_MASTER_SSID (see "LST parameters for the BQL share option" on page 234).

In case of a slave subsystem, the SSID and the BQL_MASTER_SSID are different.

In case of a master subsystem, the SSID and the BQL_MASTER_SSID are identical.

Clearing the SYNC file

If an STC or batch job terminates abnormally, the entry in the SYNC file is not removed and other subsystems are prevented from gaining control over the product database. If you want to enable other subsystems to gain access to the product database, clear the SYNC file by one of the following methods:

- Restarting the STC or rerunning the batch job

  An existing entry in the SYNC file prevents other subsystems from taking control over the database, but it does not prevent the same subsystem from regaining control. Subsystem-related information in the SYNC file is checked during restart. The entry in the SYNC file will be removed when the STC is stopped or the batch job terminates properly.

  If an STC or batch job has terminated abnormally, another subsystem from the same sysplex (same GRS complex) is allowed to gain control over the database. A GRS enqueue safeguards against several subsystems taking control of the same database.

- Running B*nn*CLSYN

  You can run job B*nn*CLSYN to clean the SYNC file (see "BnnCLSYN: Cleaning the SYNC file" on page 239).

BnnCLSYN: Cleaning the SYNC file

Overview

Program BST05SYC clears information from the SYNC file.

The program is called by the batch utility B*nn*CLSYN. JCL for B*nn*CLSYN can be found in the BETA*nn*.CNTL (replace *nn* with the number of the Beta product).

When to run B*nn*CLSYN

Run batch utility B*nn*CLSYN in the following situations:

- If you are instructed to do so by Beta Systems support
- If the following error message prevents the restart of a subsystem:

  `xxx9545E MASTER SUBSYSTEM IN USE`

  For a description of this message, see *BSA Messages and Codes*.

Before you run B*nn*CLSYN

The MASTER SUBSYSTEM IN USE message is normal if another active started task or batch job has currently control of the BQL database. Before running B*nn*CLSYN, you **must make absolutely sure** that the message is not caused by an active started task or an active batch utility.

If no active STC or batch job is using the database, the MASTER SUBSYSTEM IN USE is likely to be caused by an obsolete entry in the SYNC file, left behind by an STC or batch job that was not terminated properly.

Note on sharing databases

The SYNC file contains an entry identifying the master subsystem. This entry is deleted by B*nn*CLSYN.

If you are sharing the BQL database (LST parameter BQL_SHARE_OPTION = ALL | SPOOL), the master subsystem must be the first subsystem to be started after B*nn*CLSYN has finished.

Please see the notes on keyword BQL_SHARE_OPTION (see page 234).

Sample JCL

The JCL must specify DISP=OLD for DD SYNCFILE.

All started tasks and batch jobs using this database must be down when you run B*nn*CLSYN.

```
+-------------------------------------------------------------------+
|jobcard                                                            |
|//CLSYN    EXEC PGM=BST05SYC                                       |
|//STEPLIB  DD  DISP=SHR,                                           |
|//             DSN=BSA.LOAD                                        |
|//SYNCFILE DD  DISP=OLD,                                           |
|//             DSN=hlq.SYNC                                        |
|//SYSUDUMP DD  SYSOUT=*                                            |
+-------------------------------------------------------------------+
```

Return codes

0	The program ended normally. The SYNC file has been cleared.
16	The program ended normally. The SYNC file was empty or the specified file is not a SYNC file.
20	Read error
25	Write error
32	DD card missing or open error
36	Wrong disposition (DISP=OLD required for DD SYNCFILE)

Notes on model spool file definitions

Requesting model spool files

When a model spool file is requested, the unit where the spool file is being created will be blocked during the process of formatting. Make sure that sufficient space is available on the spool, otherwise access to all databases will be blocked during the process of formatting.

DB share / delete model spool file definitions

When database sharing is active, model spool files can no longer be deleted online, but can only be deleted using program BST05CMD.

Spool files in status ERR or FMT

Spool files in status ERR or FMT can no longer be placed online in model status, nor can they be deleted online. (The reason for the error status is most likely due to errors occurring during allocation or formatting. Check the job log of the started task).

- To delete spool files that are in status ERR or FMT, use program BST05CMD (see "BST05CMD: Updating the database definition file" on page 244).

Note: The above applies not only to file type SP - Spool, but also to the other types of spool files supported by some products (CA - Cache, GL - Global, IX - Index, SR - Reload).

JCL for BSA database batch utilities

Sample JCL

This is the JCL required to execute batch commands for the administration and maintenance of databases.

```
+-----------------------------------------------------------+
|jobcard                                                    |
|//stepname EXEC PGM=BST01RFF,REGION=0M,                    |
|//   PARM='S=nn,BnnLST=xx,PGM=yyyyyyyy,SIGNON=NO,SVC=zzz'  |
|//*                                                        |
|//STEPLIB  DD  DISP=SHR,                                   |
|//             DSN=BSA.LOAD                                |
|//         DD  DISP=SHR,                                   |
|//             DSN=BETAnn.LOAD                             |
|//SFFPARM  DD  DISP=SHR,                                   |
|//             DSN=BETA.PARMLIB                            |
|//BnnDEF   DD  DISP=SHR,                                   |
|//             DSN=hlq.DEF                                 |
|//*                                                        |
|//BQLIN        DD  *                                       |
|               command1                   -                |
|               command2                   -                |
|               commandn                                    |
|/*                                                         |
|//BQLPRINT DD  SYSOUT=*                                    |
|//SYSPRINT DD  SYSOUT=*                                    |
|//BQLSAVE  DD  DISP=SHR,                                   |
|//             DSN=dsname                                  |
|//*                                                        |
+-----------------------------------------------------------+
```

EXEC statement

```
//stepname EXEC PGM=BST01RFF,REGION=0M,
//   PARM='S=nn,BnnLST=xx,PGM=yyyyyyyy,SIGNON=NO,SVC=zzz'
```

In the PARM parameter, replace:

- *nn* with the product number
- *xx* with the identifier of the LST member
- *yyyyyyyy* with the name of the batch program
- *zzz* with the SVC number

Note: SIGNON=NO is normally required for the BSA database batch utilities.

DD statements

DD name	Description
STEPLIB	Concatenation of the BSA load library and the product load library
SFFPARM	Startup parameter library
B*nn*DEF	Database definition file of the product In the sample, replace *nn* with the product number and *hlq* with the high-level qualifier of the dataset name. **Note:** You can use a DUMMY DD statement when the program signs on to the product started task (SIGNON=YES).
BQLIN	Specifies the instructions that are to be carried out For a description of the commands, see the descriptions of the individual BSA database batch utilities. **Syntax**: The following general syntax rules apply when specifying commands under DD BQLIN: • A command can start at any column on the line. • Use the continuation sign (-) at the end of the line if the command is continued on the following line. The continuation sign must be separated from the rest of the line by at least one blank.
BQLPRINT	Log file used by the program
SYSPRINT	For programs that invoke IDCAMS: Log file used by IDCAMS
BQLSAVE	For BST05DBL Sequential file that is used by the program when unloading data from or loading data into the database

Return codes

0	The batch commands have been successfully completed.
16	An error occurred while executing the batch commands (command syntax or statement error).

Information codes

For a list of information codes, see "Database codes" in *BSA Messages and Codes*.

BST05CMD: Updating the database definition file

Overview

BST05CMD updates information in the database definition file. For example, BST05CMD can be used to delete the definitions for spool files (type SPOOL, INDEX, etc.) from the database definition file (B*nn*DEF).

Deleting spool file definitions

Please observe the following when deleting spool file definitions:

- You can delete spool files in status MOD, EMP, ERR, or FMT.
- All database activities, i.e. all STCs and batch jobs that are working with the database must be stopped beforehand.
- Parameter SIGNON=NO must be set when the program is called.
- The following command must be defined in BQLIN:

  ```
  DEFINE DELETE FILE shortname
  ```

  where *shortname* is the short name of the spool, cache or index file to be deleted.

Identifying the shortname

To find the short name *shortname* that is to be used in the Delete command in BQLIN, proceed as follows:

1. Select online option **D.1**.
2. Use line command **S** to select the definition (in status MOD, FMT, ERR or EMP) to be deleted.

   Panel "Display Data Set Definition" is called. The **Short Name** field shows the short name to be used in the DELETE command.

Return codes

0	The program ended normally. The definition has been deleted.
16	The delete command was not executed. You will find additional information in BQLPRINT.
32	BQLPRINT or the BQLIN DD statement was missing, or the database was open for another STC or batch job.

Sample JCL for DELETE

Replace *nn* with the product number and *xx* with the identifier of the LST member.

```
+-------------------------------------------------------------+
|jobcard                                                      |
|//stepname EXEC PGM=BST01RFF,REGION=0M,                      |
|//    PARM='S=nn,BnnLST=xx,PGM=BST05CMD,SIGNON=NO'           |
|//*                                                          |
|//STEPLIB  DD  DISP=SHR,DSN=BSA.LOAD                         |
|//         DD  DISP=SHR,DSN=BETAnn.LOAD                      |
|//SFFPARM  DD  DISP=SHR,DSN=BETA.PARMLIB                     |
|//BnnDEF   DD  DISP=SHR,DSN=BETAnn.DB.DEF                    |
|//*                                                          |
|//BQLIN        DD  *                                         |
|               DEFINE DELETE FILE shortname                  |
|/*                                                           |
|//BQLPRINT DD  SYSOUT=*                                      |
|//SYSPRINT DD  SYSOUT=*                                      |
|//*                                                          |
+-------------------------------------------------------------+
```

Setting spool files to OFFLINE/ONLINE

You can use BST05CMD to change the status (ONLINE/OFFLINE) of a spool file definition in the database definition file. Spool files (type SPOOL, INDEX, etc.) with status OFFLINE remain closed when the STC or a batch job is started. If you are working with a large number of files, this can considerably speed up the startup process.

- Spool files that have been set to OFFLINE status are not opened until they are explicitly accessed (BROWSE, etc.). Once accessed, they remain open for as long as the job or STC is running and are available for any inserts that may need to be made.
- It is not possible to make inserts until the spool file concerned has been opened; the INSERT procedure itself will **not** open the spool file.
- If the LST parameter BQL_OPEN_FILES=YES is set, all spool files will be opened at startup, including those that have been set to OFFLINE status.

Note the following when changing spool file status with BST05CMD:

- All database activities, i.e. all STCs and batch jobs that are working with the database must be stopped beforehand.
- Parameter SIGNON=NO must be set when the program is called.
- The following command in BQLIN sets the status to OFFLINE:

  ```
  DEFINE UPDATE FILE shortname STATUS OFFLINE
  ```

  where *shortname* is the short name of the spool file to be set to OFFLINE status.

  The following command in BQLIN sets the status to ONLINE:

  ```
  DEFINE UPDATE FILE shortname STATUS ONLINE
  ```

BST05UPF: Updating file information in the database definition file

Overview

The BST05UPF utility will not normally be used as part of data center routine, therefore we recommend that you only run BST05UPF after consultation with Beta Systems support. BST05UPF is used to update file information in an existing database definition file. For example, you may want to run this program when:

- Specifying attributes (name, size, volume, etc.) for a database component
- Changing the size of a database component
- Renaming a database component
- Changing the high water mark of a database component

Use command DEFINE UPDATE to specify the file information under DD BQLIN.

Caution

Whenever you run this program, please check the logs, even if the job ends with RC=0. Under certain circumstances, RC=0 may indicate that the definition database has not been found, and that the update process has not started for this reason.

Changing the physical dataset

Program BST05UPF only updates the information in the database definition file. Additional actions are required to change the corresponding physical dataset.

STC must be down

You must stop the started task before updating file information in the database definition file (DEFINE UPDATE FILE ...).

Syntax DD BQLIN

```
DEFINE UPDATE FILE                                      -
           filename | OLDDSN(dsname)                    -
         [ DSNAME(dsname) ]                             -
         [ MIRROR(NO)                                   -
         [ SPACE(cyl[,cyl]) ]                           -
         [ VOLUME(volume|SMS) ]                         -
         [ HWM(n) ]                                     -
         [ MAXSIZE(cyl) ]                               -
         [ BUFFER(n) ]                                  -
         [ STORAGECLASS(name) ]                         -
         [ MANAGEMENTCLASS(name) ]                      -
         [ DATACLASS(name) ]
```

Keywords DD BQLIN

DEFINE UPDATE FILE	Introduces the update command Use either *filename* or OLDDSN (*dataset name*) to specify which file to update. All other parameters are optional.
filename	filename is the short name (logical name) of the database component **Note on spool files:** When updating spool files, you must enter the name of the file that was generated using command DEFINE INSERT FILE. For example, if the number 20 was generated automatically, the file name will be SP000020.
OLDDSN(*dsname*)	dsname is the existing dataset name **Note on SYSVAR support:** If you code a dataset name instead of a short name, you must use exactly the same notation as in the data-base definition file. If the dataset name contains a static system symbol, you must specify the name of this symbol (&sysvar.).
DSNAME(*dsname*)	*dsname* is the new dataset name (for example, when renaming a database component)
MIRROR(NO)	Enter NO after the keyword if you do not want to create mirror databases.
SPACE(*cyl)*	Size of the VSAM dataset in cylinders; specify primary and secondary space when using dynamic databases: SPACE(*cyl*,*cyl*)
VOLUME(*volume* | SMS)	Volume to be used for the VSAM dataset. When SMS managed files are used, SMS must be entered rather than the volume name.
HWM(*nn*)	High water mark in percentage points (0..99)
MAXSIZE(*cyl)*	User-defined warning threshold value for file size (in cylinders); enables users to monitor database growth following automatic database extension (online under option **D.1** or via the messages xxx9549I and xxx9549W)
BUFFER(*n*)	For database components of type DATA or KEY: Number of buffers, in 4K blocks, to be used by the BQL speedmaster function If A2G_SUPPORT=NO, the maximum value is limited to 30000.
STORAGECLASS(*name*)	Name of the storage class (when using SMS managed files)
MANAGEMENTCLASS (*name*)	Name of the management class (when using SMS managed files)
DATACLASS(*name*)	Name of the data class (when using SMS managed files)

Example: Activating database mirroring

Overview

The decision whether to use software mirroring for a product database is typically taken during the installation procedure of the product. Following is a sample job that shows how you can activate software mirroring for an existing product database.

Important: All database access must be stopped during the entire procedure.

Steps

The following example refers to the _beta check|z MAIN database.

Step 1:	BST05CMD inserts the definitions for MAINKEYM (mirrored key of MAIN) and MAINDATM (mirrored data of MAIN) in the _beta check|z database definition file.
Step 2:	BST05CMD updates the definitions of MAINKEY (key of MAIN) and MAINDATA (data of MAIN) in the _beta check|z database definition file.
Step 3:	IDCAMS creates physical mirror data component.
Step 4:	IDCAMS creates physical mirror key component.
Step 5:	IDCAMS copies the content from the data/key database component to the mirror data/key database component.

JCL

```
+--------------------------------------------------------------------+
|jobcard                                                             |
|//*                                                                 |
|//* STEP 1: INSERT MAINKEYM AND MAINDATM DEFINITION IN B91DEF       |
|//UPDDEF1 EXEC PGM=BST01RFF,REGION=0M,                              |
|//             PARM=('S=91,B01LST=00,B91LST=00',                    |
|//             'PGM=BST05CMD,SIGNON=NO')                            |
|//*                                                                 |
|//STEPLIB  DD  DISP=SHR,DSN=BETA.APFLOAD                            |
|//         DD  DISP=SHR,DSN=BSA.LOAD                                |
|//         DD  DISP=SHR,DSN=BETA91.LOAD                             |
|//*                                                                 |
|//SFFPARM  DD   DISP=SHR,DSN=BETA.PARMLIB                           |
|//*                                                                 |
|//B91DEF   DD   DISP=SHR,DSN=BETA91.BQL.DEF                         |
|//*                                                                 |
|//BQLIN    DD  *                                                    |
| DEFINE INSERT FILE MAINDATM               -                        |
|        MIRRORFILE                         -                        |
|        PRODUCT B91                        -                        |
|        COMMENT ' BETA91 DEFINITION TABLES' -                       |
|        LNAME  MAIN_DATABASE_DATA_FILE     -                        |
|        DSNAME BETA91.BQL.MIRRMAIN.DATA    -                        |
|        KEY MAINKEYM                       -                        |
|        VOLUME(SMS)                        -                        |
|        UNIT   3390                        -                        |
|        SPACE  10                          -                        |
|        CISIZE 4096                                                 |
| DEFINE INSERT FILE MAINKEYM               -                        |
|        MIRRORFILE                         -                        |
|        PRODUCT B91                        -                        |
|        COMMENT ' BETA91 DEFINITION TABLES' -                       |
|        LNAME  MAIN_DATABASE_KEY_FILE      -                        |
|        DSNAME BETA91.BQL.MIRRMAIN.KEY     -                        |
|        VOLUME(SMS)                        -                        |
|        UNIT   3390                        -                        |
|        SPACE  10                          -                        |
|        CISIZE 4096                                                 |
|/*                                                                  |
|//BQLPRINT DD  SYSOUT=*                                             |
|//SYSPRINT DD  SYSOUT=*                                             |
|//*                                                                 |
|//SFFFDUMP DD  SYSOUT=*                                             |
|//SYSUDUMP DD  SYSOUT=*                                             |
|//SYSABEND DD  SYSOUT=*                                             |
|//*                                                                 |
                                                         (continued)
```

(continued)

```
|//* STEP 2: UPDATE MAINKEY AND MAINDATA DEFINITION IN B91DEF          |
|//UPDDEF2 EXEC PGM=BST01RFF,REGION=0M,                               |
|//             PARM=('S=91,B01LST=00,B91LST=00',                     |
|//             'PGM=BST05CMD,SIGNON=NO')                             |
|//*                                                                  |
|//STEPLIB  DD  DISP=SHR,DSN=BETA.APFLOAD                             |
|//         DD  DISP=SHR,DSN=BSA.LOAD                                 |
|//         DD  DISP=SHR,DSN=BETA91.LOAD                              |
|//*                                                                  |
|//SFFPARM  DD   DISP=SHR,DSN=BETA.PARMLIB                            |
|//*                                                                  |
|//B91DEF   DD   DISP=SHR,DSN=BETA91.BQL.DEF                          |
|//*                                                                  |
|//BQLIN    DD  *                                                     |
| DEFINE UPDATE FILE MAINDATA -                                       |
|        MIRROR (MAINDATM)                                            |
| DEFINE UPDATE FILE MAINKEY  -                                       |
|        MIRROR (MAINKEYM)                                            |
|/*                                                                   |
|//BQLPRINT DD  SYSOUT=*                                              |
|//SYSPRINT DD  SYSOUT=*                                              |
|//*                                                                  |
|//SFFFDUMP DD  SYSOUT=*                                              |
|//SYSUDUMP DD  SYSOUT=*                                              |
|//SYSABEND DD  SYSOUT=*                                              |
|//*                                                                  |
|//* STEP 3: CREATE PHYSICAL MIRROR DATA COMPONENT                    |
|//DBDEF1  EXEC PGM=IDCAMS                                            |
|//SYSIN    DD  *                                                     |
| DEFINE       CLUSTER(                                           -   |
|                 NAME(BETA91.BQL.MIRRMAIN.DATA)                  -   |
|             CYLINDERS(40,5)                                     -   |
|          SHAREOPTIONS(3 3)                                      -   |
|                 OWNER(BETA91)                                   -   |
|                UNIQUE                                           -   |
|           NONINDEXED                                            -   |
|                      )                                          -   |
|                  DATA(                                          -   |
|                  NAME(BETA91.BQL.MIRRMAIN.DATA.DATA)            -   |
|   CONTROLINTERVALSIZE(4096)                                     -   |
|           RECORDSIZE(4088 4088)                                 -   |
|                    )                                                |
|/*                                                                   |
|//SYSPRINT DD  SYSOUT=*                                              |
|//*                                                                  |
```

(continued)

(*continued*)

```
|//* STEP 4: CREATE PHYSICAL MIRROR KEY COMPONENT                       |
|//DBDEF2  EXEC PGM=IDCAMS                                              |
|//SYSIN    DD  *                                                       |
| DEFINE       CLUSTER(                                             -   |
|                 NAME(BETA91.BQL.MIRRMAIN.KEY)                     -   |
|             CYLINDERS(20,5)                                       -   |
|          SHAREOPTIONS(3 3)                                        -   |
|                 OWNER(BETA91)                                     -   |
|                UNIQUE                                             -   |
|            NONINDEXED                                             -   |
|                      )                                            -   |
|                  DATA(                                            -   |
|                  NAME(BETA91.BQL.MIRRMAIN.KEY.DATA)               -   |
|   CONTROLINTERVALSIZE(4096)                                       -   |
|            RECORDSIZE(4088 4088)                                  -   |
|                     )                                                  |
|/*                                                                     |
|//SYSPRINT DD  SYSOUT=*                                                |
|//*                                                                    |
|//* STEP 5: COPY DATA FROM DATA/KEY TO MIRROR DATA/KEY                 |
|//DBREPRO EXEC PGM=IDCAMS                                              |
|//*                                                                    |
|//OLD1     DD  DISP=OLD,DSN=BETA91.BQL.MAIN.DATA                       |
|//NEW1     DD  DISP=SHR,DSN=BETA91.BQL.MIRRMAIN.DATA                   |
|//OLD2     DD  DISP=OLD,DSN=BETA91.BQL.MAIN.KEY                        |
|//NEW2     DD  DISP=SHR,DSN=BETA91.BQL.MIRRMAIN.KEY                    |
|//*                                                                    |
|//SYSPRINT DD  SYSOUT=*                                                |
|//SYSIN    DD  *                                                       |
| REPRO  INFILE(OLD1) OUTFILE(NEW1)                                     |
| REPRO  INFILE(OLD2) OUTFILE(NEW2)                                     |
|/*                                                                     |
+-----------------------------------------------------------------------+
```

Example: Deactivating database mirroring

Overview

The decision whether to use software mirroring for a product database is typically taken during the installation procedure of the product. Following is a sample job that shows how you can deactivate software mirroring for an existing product database.

Important: All database access must be stopped during the entire procedure.

Steps

The example refers to the _beta check|z MAIN database.

Step 1: BST05UPF updates the definitions of MAINDATA (data of MAIN) and MAINKEY (key of MAIN) in the _beta check|z database definition file.

Step 2: BST05UPF deletes the definitions for MAINDATM (mirrored data of MAIN) and MAINKEYM (mirrored key of MAIN) in the _beta check|z database definition file.

Step 3: IDCAMS deletes the physical mirror database components.

JCL

```
+---------------------------------------------------------------------+
|jobcard                                                              |
|//*                                                                  |
|//* STEP 1: UPDATE MAINKEY AND MAINDATA DEFINITION IN B91DEF         |
|//UPDDEF1 EXEC PGM=BST01RFF,REGION=0M,                               |
|//             PARM=('S=91,B01LST=00,B91LST=00',                     |
|//             'PGM=BST05UPF,SIGNON=NO')                             |
|//*                                                                  |
|//STEPLIB  DD  DISP=SHR,DSN=BETA.APFLOAD                             |
|//         DD  DISP=SHR,DSN=BSA.LOAD                                 |
|//         DD  DISP=SHR,DSN=BETA91.LOAD                              |
|//*                                                                  |
|//SFFPARM  DD   DISP=SHR,DSN=BETA.PARMLIB                            |
|//*                                                                  |
|//B91DEF   DD   DISP=SHR,DSN=BETA91.BQL.DEF                          |
|//*                                                                  |
|//BQLIN    DD  *                                                     |
|  DEFINE UPDATE FILE MAINDATA -                                      |
|         MIRROR (NO)                                                 |
|  DEFINE UPDATE FILE MAINKEY  -                                      |
|         MIRROR (NO)                                                 |
|/*                                                                   |
|//BQLPRINT DD  SYSOUT=*                                              |
|//SYSPRINT DD  SYSOUT=*                                              |
|//*                                                                  |
|//SFFFDUMP DD  SYSOUT=*                                              |
|//SYSUDUMP DD  SYSOUT=*                                              |
|//SYSABEND DD  SYSOUT=*                                              |
|//*                                                                  |
|//* STEP 2: DELETE MAINKEYM AND MAINDATM DEFINITION IN B91DEF        |
|//UPDDEF2 EXEC PGM=BST01RFF,REGION=0M,                               |
|//             PARM=('S=91,B01LST=00,B91LST=00',                     |
|//             'PGM=BST05UPF,SIGNON=NO')                             |
|//*                                                                  |
|//STEPLIB  DD  DISP=SHR,DSN=BETA.APFLOAD                             |
|//         DD  DISP=SHR,DSN=BSA.LOAD                                 |
|//         DD  DISP=SHR,DSN=BETA91.LOAD                              |
|//*                                                                  |
|//SFFPARM  DD   DISP=SHR,DSN=BETA.PARMLIB                            |
|//*                                                                  |
|//B91DEF   DD   DISP=SHR,DSN=BETA91.BQL.DEF                          |
|//*                                                                  |
|//BQLIN    DD  *                                                     |
|  DEFINE DELETE FILE MAINDATM                                        |
|  DEFINE DELETE FILE MAINKEYM                                        |
|/*                                                                   |
|//BQLPRINT DD  SYSOUT=*                                              |
|//SYSPRINT DD  SYSOUT=*                                              |
|//*                                                                  |
|//SFFFDUMP DD  SYSOUT=*                                              |
|//SYSUDUMP DD  SYSOUT=*                                              |
|//SYSABEND DD  SYSOUT=*                                              |
|//*                                                                  |
|//* STEP 3: DELETE THE PHYSICAL MIRROR DATABASE                      |
|//DELETE  EXEC PGM=IDCAMS                                            |
|//SYSPRINT DD  SYSOUT=*                                              |
|//SYSIN    DD  *                                                     |
| DELETE BETA91.BQL.MIRRMAIN.DATA                                     |
| DELETE BETA91.BQL.MIRRMAIN.KEY                                      |
|/*                                                                   |
+---------------------------------------------------------------------+
```

BST05LAU: Locking and unlocking a BQL database

Overview

Program BST05LAU is used to lock and to unlock a BQL database.

Lock the BQL database when you want to stop all access to the database, but do not want to stop the started task. Unlock the database to make it available again.

This feature is useful, for example, when backing up a database using the concurrent copy function. Locking the database while the concurrent copy environment is initialized ensures the consistency of the database copy.

Caution

Depending on the product, locking the database may not be enough to ensure the consistency of the database backup. **Special conditions** may apply and **additional steps** may be necessary. Always refer to the product documentation first before using the concurrent copy function.

Syntax Parm parameter

`SSID=`*`ssid`*`,STAT=LOCK | UNLOCK`

where *ssid* is the subsystem ID that the LOCK or UNLOCK applies to.

Sample JCL

```
+-----------------------------------------------------------------+
|//BST05LAU JOB BETA,MSGCLASS=P,CLASS=A,NOTIFY=&SYSUID            |
|//BST05LAU EXEC PGM=BST05LAU,REGION=0M,                          |
|//             PARM='SSID=ssid,STAT=LOCK'                        |
|//*                                                              |
|//STEPLIB  DD  DISP=SHR,                                         |
|//             DSN=BETA.APFLOAD               <=== BSA APF LIBRARY  |
|//         DD  DISP=SHR,                                         |
|//             DSN=BSA.LOAD                   <=== BSA LOAD LIBRARY |
|//                                                               |
+-----------------------------------------------------------------+
```

Note

BST05LAU requires SIGNON = YES, which is the default.

Loading data into a BQL database (BST05DBL)

Introduction

Program BST05DBL does the following:

- It loads data from a sequential file into a BQL database.
- It unloads data from a BQL database to a sequential file.

Use the command LOAD under DD BQLIN to load data into a BQL database.

Note

During the execution of this command, the BQL does not accept any other requests.

STC must be down

You must stop the started task before loading data into the BQL database using BST05DBL.

Structure of sequential file

DD BQLSAVE defines the sequential file containing the data to be loaded into the database. This file must be variable blocked.

The header of the records must have this structure:

Contents	Position	Length	Format
Record length	0	2	fixed
Reserved for future use	2	2	fixed 0
Table name	4	8	character
Reserved for future use	12	4	character

- The first four bytes contain the record descriptor word (RDW) of a variable record. The existence of the record descriptor word is a prerequisite for assembling a variable record, according to z/OS.
- The contents of the table must be entered in internal format below the record header (according to the table names entered).
- The contents of the table entered here must correspond to the entries in the definition database.

Syntax DD BQLIN

```
+-------------------------------------------------------+
|LOAD TABLE * | tablename,...                         -|
|            DDNAME(ddname) | DSNAME(dsname)          -|
|                              MEMBER(name)           -|
|                              VOLUME(volume)         -|
|                              UNIT(unit)             -|
|            GENFILE(nn)                               |
+-------------------------------------------------------+
```

Keywords DD BQLIN

Keyword	Description
LOAD	Introduces the load command
TABLE * TABLE *tablename*,...	Specifies which of the data contained in the sequential file should be loaded into the tables of the database Code one or several table names to load only records of the specified tables. Code an asterisk (*) to load the records of all tables.
DDNAME(*ddname*)	DD name of the sequential file
DSNAME(*dsname*)	Dataset name of the sequential file
MEMBER(*name*)	Name of the member (if *dsname* is a PO dataset)
VOLUME(*volume*)	Volume where the sequential file is stored (if not cataloged)
UNIT(*unit*)	Unit where the sequential file can be accessed (if not cataloged)
GENFILE(*nn*)	Number of the generation file **Note**: GENFILE(*nn*) is required if the file to be loaded is a GEN file.

Unloading data from a BQL database (BST05DBL)

Introduction

Program BST05DBL does the following:

- It loads data from a sequential file into a BQL database.
- It unloads data from a BQL database to a sequential file.

Use the command UNLOAD under DD BQLIN to unload data to a sequential file. The sequential file is defined in the JCL by DD BQLSAVE.

Note: During the execution of this command, the BQL does not accept any other requests.

STC must be down

You must stop the started task before loading data into the BQL database using BST05DBL.

Syntax DD BQLIN

```
UNLOAD  TABLE tablename,... | FILE filename
```

Keywords DD BQLIN

UNLOAD	Introduces the unload command Use either **TABLE *tablename*** to specify the name of one or several tables to unload or **FILE *filename*** to unload all tables of this file.
TABLE *tablename*,...	Specifies the name of up to 10 tables whose contents are to be unloaded to DD BQLSAVE
FILE *filename*	Specifies the name of the database component whose contents are to be unloaded to DD BQLSAVE

Formatting a database component (BST05FOR)

Introduction

The program BST05FOR formats the unformatted space of a database component.

If the specified dataset does not yet exist (for example, spool file of type model or installation with new database), BST05FOR invokes IDCAMS internally to allocate the dataset first and then formats it.

Note: Because IDCAMS is invoked internally, the DD card SYSPRINT must be coded in the JCL of the batch job or in the JCL of the STC.

Formatting added space

When you enlarge a database component that already contains valid data, only the added space is formatted.

Note: The keyword EXTEND, which was previously coded when formatting added space, is now obsolete and will be ignored if present.

Specifying volume, space etc.

BST05FOR uses the values that are stored in the database definition file when it invokes IDCAMS to allocate a new database component. You can also code values using the appropriate BQLIN keywords, which will override the corresponding values of the database definition file.

When you allocate and format the database definition file, you must code the keyword DEFFILE and specify the space, volume etc. using the BQLIN keywords.

Syntax DD BQLIN

```
+-----------------------------------------------------------+
|FORMAT FILE                                               -|
|         SNAME(shortname) | DSNAME dsname | DSN dsname     -|
|       [ CISIZE(n) | CSI(n) ]                             -|
|       [ VOLUME(volume) ]                                  -|
|       [ SPACE(cyl[,cyl]) ]                                -|
|       [ STORAGECLASS(name) ]                              -|
|       [ MANAGEMENTCLASS(name) ]                           -|
|       [ DATACLASS(name) ]                                 -|
|       [ DEFFILE ]                                         -|
|       [ NOSPEED ]                                         |
+-----------------------------------------------------------+
```

Keywords DD BQLIN

FORMAT FILE	Introduces the format command Use either SNAME(*shortname*), DSNAME *dsname*, or DSN *dsname* to specify which database component to format.
SNAME(*shortname*)	Short name (logical name) of the database component
DSNAME *dsname* DSN *dsname*	Dataset name of the database component **Note on SYSVAR support:** If you code a dataset name instead of a short name, you must use exactly the same notation as in the database definition file. If the dataset name contains a static system symbol, you must specify the name of this symbol (&sysvar.).

CISIZE(*n*)
CSI(*n*)
	Control interval size of the VSAM dataset
	n must be a multiple of 4096, and it must be <= 28672 (28K).
	Note: The recommended CIsize is 4096 (4K). Because 4K is the standard value, the documentation often uses "number of 4K blocks" when a value refers to the CIsize.

VOLUME(*volume*|SMS)	Volume to be used for the VSAM dataset. When SMS managed files are used, SMS must be entered rather than the volume name.
SPACE(*n*) | SPACE(*n,n)*	Size of the VSAM dataset in cylinders For example, SPACE(10) means that the primary space is 10 cyl. SPACE(10,5) means that 10 cyl will be used as primary space, and 5 cyl will be used as secondary space if the database is extended (see "Dynamic database extension" on page 224).
STORAGECLASS(*name*)	Name of the storage class (when using SMS managed files)
MANAGEMENTCLASS (*name*)	Name of the management class (when using SMS managed files)
DATACLASS(*name*)	Name of the data class (when using SMS managed files)
DEFFILE	This keyword must be coded when formatting the database definition file.
EXTEND	This keyword is obsolete.
NOSPEED	When this keyword is present, the dataset will be preformatted by VSAM (which means that performance will be slower and formatting will take longer). As default, this parameter is inactive.

Building or rebuilding keys for a table (BST05DBL)

Introduction

You can use the program BST05DBL to build or rebuild the keys of one or several tables of the BQL database.

Note

During the execution of this command, the BQL does not accept any other requests.

STC must be down

You must stop the started task before building or rebuilding the keys using BST05DBL.

Syntax DD BQLIN

```
+-------------------------------------------------------+
|ADDKEY keyname | tablename | FILE  filename           -|
|       GENFILE(nn)                                    -|
|       CLEAR                                           |
+-------------------------------------------------------+
```

Keywords DD BQLIN

ADDKEY	Specify which keys to build: ADDKEY *keyname* builds this key ADDKEY *tablename* builds all keys for this table ADDKEY FILE *filename* builds all keys of this file
GENFILE(nn)	Number of the generation file
CLEAR	Specify this keyword to delete the existing contents of a key before building new keys.

Deleting the contents of a table, key, or file (BST05DBL)

Introduction

Use program BST05DBL to delete the contents of files, tables, and keys in the BQL database

Note

- This command is executed without any checking for the formal correctness of the command or whether the table in question allows for the usage of this command, or whether an entry for a key is missing, for example.
- During the execution of this command, the BQL does not accept any other requests.

STC must be down

You must stop the started task before building or rebuilding the keys using BST05DBL.

Syntax DD BQLIN

- To delete the contents of one or several keys:

```
+----------------------------------------------------+
|DELKEYSEQ keyname,... | tablename,... | FILE filename -|
|        [ GENFILE(nn) ]                              |
+----------------------------------------------------+
```

- To delete the contents of a table:

```
+----------------------------------------------------+
|DELTABSEQ  tablename [ GENFILE(nn) ]                 |
+----------------------------------------------------+
```

- To delete the contents of a file:

```
+----------------------------------------------------+
|CLEAR FILE filename [ GENFILE(nn) ]                  |
+----------------------------------------------------+
```

BQLIN Keywords

CLEAR DELTABSEQ DELKEYSEQ	Introduces the command: CLEAR FILE *filename* deletes the contents of all tables and all keys of this file DELTABSEQ *tablename* deletes the contents of this table DELKEYSEQ *keyname*,... deletes the contents of the specified key or keys DELKEYSEQ *tablename* deletes the contents of the keys of this table DELKEYSEQ FILE *filename* deletes the contents of the keys of all tables of this file
GENFILE(*nn*)	Number of the generation file

BST05ANA: Database analysis and repair utility

Overview

BST05ANA is a BSA database utility which you can use to:

- Analyze spool file contents, for example, to find out which indexes and lists are contained (entirely or in part) in a spool file (CHECK command)

  _beta doc|z plus administrators also use the CHECK command with the subcommand BUILD(SFR) to enable full recovery for spool files with list or index fragments that have been read in before Beta 93 Fast Retrieval V4R1. This command runs once against all existing spool files of the type SPOOL and INDEX during the upgrade procedure to V4 or later.
- Reorganize spool files in order to release used blocks that contain no valid data (REPAIR command)
- Reload all data from the archive that is affected by spool file corruption (RELOAD command)
- Empty spool files by moving all their data to other spool files (MOVE command)

Command availability matrix

The functions of BST05ANA are not available for all products and spool file types. The following table shows whether the corresponding command is available (+) or not (-).

Product and spool file type		CHECK	CHECK BUILD (SFR)	MOVE	RELOAD	REPAIR
Beta 92	Type SPOOL	+	-	-	-	-
Beta 93	Type SPOOL	+	+	-	+	-
Beta 93	Type RELOAD	+	+	-	+	-
Beta 93 FR	Type CACHE	+	-	-	-	+
Beta 93 FR	Type GLOBL	+	+	-	-	-
Beta 93 FR	Type INDEX	+	+	-	-	+
Beta 93 FR	Type SPOOL	+	+	+	+	+
Beta 93 FR	Type RELOAD	+	+	+	+	+

License check

The following commands check for the presence of a valid license for the corresponding product:

	CHECK	CHECK BUILD (SFR)	MOVE	RELOAD	REPAIR
License article			`I06`		`I04`

Running BST05ANA in slave or master mode

You can run the utility BST05ANA in slave mode (SIGNON=YES) for almost all of the functions that are described in this section.

Exception: Spool file reorganization

BST05ANA must run in master mode (SIGNON=NO) when reorganizing spool files. This means that you must stop the _beta doc|z plus started tasks before running the batch utility with the following command:

```
REPAIR DSNAME(dsname) RECORD(REO) DO-IT
```

The started tasks can remain active when this function runs in analyze mode (same command without **DO-IT**).

JCL

```
+----------------------------------------------------------------------+
|jobcard                                                               |
|//BST05ANA EXEC PGM=BST01RFF,REGION=0M,PARM=('S=nn',                  |
|//             'PGM=BST05ANA',                                        |
|//             'B01LST=xx',                                           |
|//             'BnnLST=xx',                                           |
|//             'SIGNON=YES')                                          |
|//*                                                                   |
|//STEPLIB  DD  DISP=SHR,                                              |
|//             DSN=BSA.LOAD                                           |
|//         DD  DISP=SHR,                                              |
|//             DSN=BETAnn.LOAD                                        |
|//*                                                                   |
|//SFFPARM  DD  DISP=SHR,                                              |
|//             DSN=BETA.PARMLIB                                       |
|//BnnDEF   DD  DISP=SHR,                                              |
|//             DSN=BETAnn.DB.DEF                                      |
|//*                                                                   |
|//BQLPROT  DD  SYSOUT=*                                               |
|//BQLLOG   DD  SYSOUT=*                                               |
|//BQLIN    DD  *                                                      |
| commands                                                             |
|/*                                                                    |
+----------------------------------------------------------------------+
```

DD statements

All of the following cards are required.

DD name	Description
BQLPROT	Contents of the spool file(s) type SPOOL analyzed **Notes**: • The presence of this DD card is checked at program start. This card must be present, even if the list of files to be analyzed does not include a spool file of type SPOOL. • The keyword EXTENDED must be coded if you want to obtain this type of report for spool files of type INDEX.
BQLLOG	Processing log of the program; it contains a log of the BQLIN command(s), a list of the datasets that were analyzed, and internal information on these datasets
BQLIN	Input statements (commands that are to be carried out by BST05ANA) Each command must be coded on a separate line. You can specify wildcards in the dataset name if the task is to be carried out for all matching datasets. Supported commands are described below with the individual tasks.

Displaying spool contents (CHECK)

Availability: All spool file types of all products
(valid license for article T04 or I04 for corresponding product must be present)

Specify the following command to display the contents of a spool file:

```
CHECK DSNAME(dsname)
```

The program examines the contents of the specified spool file. If at least one page of a list is stored in the spool file under analysis, this list is displayed in DD BQLPROT.

EXTENDED to create BQLPROT for INDEX spools

Availability: _beta doc|z plus spool files of type INDEX

By default, BST05ANA does not create the BQLPROT report when analyzing spool files of the type INDEX because of performance reasons. Code the keyword EXTENDED if you need this information, for example:

```
CHECK DSNAME(BETA97.INDEX*) EXTENDED
```

BUILD(SFR) to enable spool file recovery

Availability: _beta doc|z plus spool files of type GLOBL, INDEX and SPOOL

_beta doc|z plus administrators also use the CHECK command with the subcommand BUILD(SFR) to enable full recovery for spool files with list or index fragments that have been read in before Beta 93 Fast Retrieval V4R1. This command runs once against all existing spool files of the type SPOOL and INDEX during the upgrade procedure to V4 or later, for example:

```
CHECK DSNAME(BETA97.SPOOL*) BUILD(SFR)
CHECK DSNAME(BETA97.INDEX*) BUILD(SFR)
```

BST05ANA writes an SFR record for each list or index fragment for which there is no pointer in the corresponding generation record of the database. This affects lists and indexes that span multiple spool files.

Emptying spool file (MOVE)

Availability: _beta doc|z plus spool files of type SPOOL
(valid license for article T06 or I06 must be present)

Specify the following command to empty a read-only spool file by moving all its lists to other spool files:

```
MOVE DSNAME(dsname)
```

The program examines the contents of the specified spool file. It copies each list to another spool file and deletes this list in the original spool file. If a list spans multiple spool files, the entire list is moved. All list fragments are deleted from the source spool files affected.

Notes

- The dataset status of the spool file must be READONLY.
- The MOVE command cannot be run in analyze mode, but you can run the CHECK command instead to find out which lists would be moved.

Reloading spool contents (RELOAD)

Availability: _beta doc|z plus spool files of type SPOOL

Specify the following command to reload the contents of a spool file from the archive, for example, because the spool file in question has become corrupted:

```
RELOAD DSNAME(dsname) DO-IT
```

The program carries out the following processing steps:

1. It examines the appropriate database tables to find out which data is stored in the specified spool files.
2. If the spool file under analysis is of the type SPOOL, it updates the generation records of all lists that are stored (in part or entirely) on this spool file as follows:
    - If a list is archived, it is marked as offline and to be reloaded.
    - If a list is not archived, it is marked as non-viewable and to be deleted.

   If the spool file under analysis is of the type INDEX, it updates the generation records of all indexes that are stored (in part or entirely) on this spool file as follows:
    - If an index is archived, it is marked as offline.
    - If an index is not archived, it is marked as offline; the corresponding list is marked as non-viewable and to be deleted.

For more detailed instructions on restoring spool content from the archive, see "Recovering or migrating spool contents" in *_beta doc|z plus Administrator Guide*.

Important requirements to enable spool file recovery:

To be able to recover the entire data of a corrupt spool file requires the following:

- The spool file must support complete recovery.

  An SFR record must be available for each list or index fragment for which there is no pointer in the corresponding generation record of the database. SFR records are generated automatically when lists and indexes are written as of Beta 93 Fast Retrieval V4R1. SFR records for earlier data in spools are generated when you run the program BST05ANA using the following command:

  ```
  CHECK DSNAME(dsname) BUILD(SFR)
  ```
- All data of this spool file must have been archived.

  If an affected list has not been archived, the program BST05ANA will mark the list for deletion and flag it as non-viewable. The remaining data of the list and its indexes will be removed from the database at the next run of the online cleanup batch utility (B97DEONL). You can recognize these lists by the value **NV** in the **Status** column of the report that is written to DD BQLPROT.

Analyze mode

If you run this command without the DO-IT parameter, the program runs in analyze mode. In this mode, the program processes all data and writes all logs, but does not actually mark any lists for reloading:

```
RELOAD DSNAME(dsname)
```

Reorganizing spool file (REPAIR)

Availability: _beta doc|z plus spool files of type SPOOL
(valid license for article I04 must be present)

Specify the following command to analyze and reorganize a spool file:

```
REPAIR DSNAME(dsname) RECORD(REO) DO-IT
```

During reorganization, blocks containing no data or invalid data are released.

Important: This command requires that BST05ANA runs in master mode (SIGNON=NO). This means that you must stop all Beta Systems Architecture started tasks before starting the spool file reorganization.

Analyze mode

If you run this command without the DO-IT parameter, the program runs in analyze mode. In this mode, the program processes all data and writes all logs, but does not actually release any data:

```
REPAIR DSNAME(dsname) RECORD(REO)
```

Running the utility in analyze mode is useful if you want to find out how many spool file blocks would become available by reorganizing the spool files. BST05ANA can run in slave mode (SIGNON=YES) in analyze mode.

Example: DD BQLPROT

The contents of the analyzed spool files (type SPOOL only) is listed under DD BQLPROT. The following information is printed for each list:

- Name (Form, extension, report)
- List date and time
- Number of pages (according to database)
- Status

  NV appears in this column if the program BST05ANA has marked a list for deletion and flagged it as non-viewable (see RELOAD command).

```
+-------------------------------------------------------------------------------+
|START  DSNAME : QAB97.Q97V.DB.SPOOL1                                            |
|                                                                                |
|FORM      EXT              REPORT           B97DATE    B97TIME      PAGES   STATUS  |
| ---------------------------------------------------------------------------------  |
|                                                                                |
|REJ       TRADE                            09.12.2009 12:08:52:74  0000009          |
|TC9761    NOARCH                           27.11.2009 11:34:18:85  0001811          |
|TC9761    FBM156                           27.11.2009 11:34:21:46  0000003          |
|TC9761    VBA156                           27.11.2009 11:34:21:85  0000012          |
|TC9761    DISK10000                        27.11.2009 11:34:22:21  0000012          |
|TC9761    SECIDXNOITEM                     27.11.2009 11:34:24:56  0000026          |
|TC9761    ITEMPHONE                        27.11.2009 11:34:24:95  0000002          |
|TC9761    AFPDVZ                           27.11.2009 11:34:25:44  0000039          |
|TC9761    AFPTLE                           27.11.2009 11:34:26:04  0000009          |
|TC9761    VBM135GT32K                      27.11.2009 11:34:26:40  0001050          |
+-------------------------------------------------------------------------------+
```

Example: DD BQLLOG

The processing log of the BST05ANA is written to DD BQLLOG; it contains a log of the BQLIN command(s), a list of the datasets that were analyzed, and internal information on these datasets.

```
+------------------------------------------------------------------------+
|CHECK DSNAME(QAB97.Q97V.DB.SPOOL1)                               00280005|
|                                                                        |
|                                                                        |
|START  CHECK   FOR FILE :  QAB97.Q97V.DB.SPOOL1                         |
|                                                                        |
|SIR RBA : 00000000  BLOCKS ALLOCATED : 03179  WITH : 00000 READER BLOCKS |
|SIR RBA : 00000FA0  BLOCKS ALLOCATED : 01415  WITH : 00000 READER BLOCKS |
|SIR RBA : 00001F40  BLOCKS ALLOCATED : 00002  WITH : 00000 READER BLOCKS |
|                                                                        |
|TIMESTAMPS  FOUND : 00000038 TIMESTAMPS  IN USE : 00000001              |
|                                                                        |
+------------------------------------------------------------------------+
```

Return codes

0	The program was terminated normally.
16	An error has occurred, for example:

- Missing DD statement
(BQLPROT, BQLLOG and BQLIN are required)
- Invalid command in DD BQLIN
- Spool file reorganization (REPAIR DSNAME(*dsname*) RECORD(REO) DO-IT) was started in slave mode (SIGNON=YES)
- License error (for example MOVE: license article I06/T06 for current product unavailable)
- Spool type or dataset status error (for example MOVE: spool file is not of type SPOOL and/or dataset status is not READONLY)

BST08OCP: Copying/Checking/Repairing/Restoring archive files

Overview

BST08OCP is an archive utility that can be used for the following tasks:

- Checking archive data, for example, checking the contents of archive datasets (CHECK command)
- Creating copies of archive datasets (COPY command)
- Updating the appropriate database tables according to the contents of the archive datasets (REPAIR command)
- Reloading data from archive datasets (RESTORE command)

The product STC must be active when BST08OCP is running.

Function support is product-dependent

BST08OCP supports different functions for different Beta Systems products. For more detailed information on supported functions, see the documentation of your product.

The following examples are taken from _beta doc|z plus.

License check

No license is necessary for the CHECK function. For all the other functions, the program verifies that the required license is present before it executes the specified function.

Sample JCL

```
+----------------------------------------------------------------------+
|jobcard                                                               |
|//BST08OCP EXEC PGM=BST01RFF,REGION=0M,PARM=('S=97',                  |
|//             'PGM=BST08OCP',                                        |
|//             'B01LST=nn',                                           |
|//             'B97LST=nn',                                           |
|//             'SIGNON=YES')                                          |
|//*                                                                   |
|//STEPLIB  DD  DISP=SHR,                                              |
|//             DSN=BETA.APFLOAD                <=== BETA APF LIBRARY   |
|//         DD  DISP=SHR,                                              |
|//             DSN=BSA.LOAD                    <=== BSA LOAD LIBRARY   |
|//         DD  DISP=SHR,                                              |
|//             DSN=BETA97.LOAD                 <=== PRODUCT LOAD LIBRARY |
|//*                                                                   |
|//SFFPARM  DD  DISP=SHR,                                              |
|//             DSN=BETA.PARMLIB                <=== LST PARAMETER      |
|//*                                                                   |
|//B97DEF   DD  DISP=SHR,                                              |
|//             DSN=BETA97.DB.DEF                                      |
|//*                                                                   |
|//IRMPRINT DD  SYSOUT=*                                               |
|//IRMPROT  DD  SYSOUT=*                                               |
|//IRMLOG   DD  SYSOUT=*                                               |
|//IRMDEL   DD  SYSOUT=*                                               |
|//IRMIN    DD  *                                                      |
|   CHECK DSNAME(BETA97.TAPE365.VTS.E04251.C002) ORDER(1)              |
|/*                                                                    |
+----------------------------------------------------------------------+
```

Return codes

0 The program terminated normally.

4 The program terminated with warnings, which can be caused by, for example:

- No data found to process

8 Inconsistencies were found in the archive data.

16 This return code can occur due to several reasons, for example:

- An invalid keyword was coded.
- A required DD statement was not coded.
- The program was unable to find a matching record for the dataset specified. (The dataset may have already expired or its name may have been misspelled.)

DD statements

The sample JCL and the following table use DD cards with prefix IRM, which is the prefix used by _beta doc|z plus. Instead of the product prefix, you can also use the prefixes BSS or BST, i.e. BSTPRINT/BSSPRINT, BSTPROT/BSSPROT, etc.

DD name	**Description**
IRMPRINT	Logs the start, the command executed, and the result
IRMPROT	Result of the analysis, for example, contents of the archive datasets analyzed This DD statement is required when using the CHECK command. It is optional for all other commands. Don't code this DD statement if you don't need this information.
IRMLOG	Used by the program to log the input statements coded in DD IRMIN and to output messages
IRMERR	Contains lists where errors have been found (these lists are also logged in IRMPROT)
IRMDEL	List of archive datasets that can be deleted; you can use DD IRMDEL as input for IDCAMS to delete these datasets
IRMIN	Input statements This is where you code the commands that are to be processed by BST08OCP.

Using concurrent copy to back up a running BQL database

Overview

BSA supports concurrent copy. This means that you can create a backup of the BQL database while the database is running.

Requirements

To back up a BQL database while it is running requires the concurrent copy function of z/OS DFSMS, or equivalent functions if you are using non-IBM products.

For detailed information on hardware requirements, refer to the appropriate IBM publication if you are using z/OS DFSMS. If you are using non-IBM products, refer to the documentation of these products.

What is concurrent copy?

Concurrent copy is a 3990 extended function that enables data center operations staff to take a copy or a dump of data while applications are updating that data. You can use concurrent copy to back up any data that can be backed up using DFSMSdss or DFDSS because DFSMSdss is the external interface to concurrent copy.

Concurrent copy delivers a copy of the data in a consistent form. The point in time of data consistency is the time when the concurrent copy environment is initialized. During the initialization of the concurrent copy environment the database is unavailable, but this takes only a matter of seconds. After the initialization of the concurrent copy environment is complete, the copy is logically complete and the database becomes available again.

While concurrent copy copies the database, all write operations to the database are intercepted. Before carrying out updates to tracks that have not yet been backed up, concurrent copy makes a copy of the original track in a side file. Backup uses the track in the side file instead of the updated track. The copy is physically complete once the concurrent copy process finishes copying the data to the output device.

Lock BQL DB and initialize CC environment

During the initialization of the concurrent copy (CC) environment the database is unavailable.

To prevent access to the database during this phase, the database must be locked. After the initialization of the concurrent copy environment is complete (indicated by message ADR734I), the database can be unlocked.

You can lock and unlock the BQL database using the program BST05LAU (see "BST05LAU: Locking and unlocking a BQL database" on page 254) or BST05CMD (see example below).

BQL DB backup using concurrent copy

The following is a description of the basic procedure for using concurrent copy to back up a BQL database. For each individual Beta Systems product, **special conditions** may apply and **additional steps** may be necessary.

Caution: Before using the concurrent copy function, always refer to the product documentation first to find out whether the product supports this function.

Procedure

To back up a BQL database using concurrent copy:

1. Run a backup job that locks the BQL database and then copies the datasets using concurrent copy function of ADRDSSU.

   Step 1: Locking the BQL database

   ```
   +--------------------------------------------------------------+
   |//stepname EXEC PGM=BST01RFF,REGION=0M,                       |
   |//    PARM='S=nn,BnnLST=xx,PGM= BST05CMD,SIGNON=YES'          |
   |// ...                                                        |
   |//BQLIN DD *                                                  |
   |   LOCK DATABASE                                              |
   +--------------------------------------------------------------+
   ```

   As a result of this step, the database components are locked. (This applies also to shared databases.)

   Step 2: Copying datasets using CC function of ADRDSSU

   ```
   +--------------------------------------------------------------+
   |//stepname EXEC PGM=ADRDSSU                                   |
   |// ...                                                        |
   |//SYSIN DD *                                                  |
   |   statement CC                                               |
   +--------------------------------------------------------------+
   ```

   As a result of this step, the concurrent copy environment is initialized (DD SYSIN specifies CONCURRENT or CC).

2. After the initialization of the concurrent copy environment is complete (Message `ADR734I (`*`xxx`*`)-`*`mmmmm`*`(`*`yy`*`),` *`date time`* `CONCURRENT COPY INITIALIZATION [ UN ] SUCCESSFUL FOR ...`), run a job to unlock the database.

   ```
   +--------------------------------------------------------------+
   |//stepname EXEC PGM=BST01RFF,REGION=0M,                       |
   |//    PARM='S=nn,BnnLST=xx,PGM= BST05CMD,SIGNON=YES'          |
   |// ...                                                        |
   |//BQLIN DD *                                                  |
   |   UNLOCK DATABASE                                            |
   +--------------------------------------------------------------+
   ```

   As a result of this job, the database components become available again.

Sample backup job

```
+----------------------------------------------------------------+
|jobcard                                                         |
|//*-------------------------------------------------------------|
|//*       STEP1: LOCK ALL DATABASES                            *|
|//*-------------------------------------------------------------|
|//*                                                             |
|//STEP1   EXEC PGM=BST01RFF,REGION=0M,                          |
|//            PARM='S=nn,BnnLST=xx,PGM=BST05CMD,SIGNON=YES'     |
|//STEPLIB  DD  DISP=SHR,DSN=BSA.LOAD                            |
|//         DD  DISP=SHR,DSN=BETAnn.LOAD                         |
|//SFFPARM  DD  DISP=SHR,DSN=BETA.PARMLIB                        |
|//BnnDEF   DD  DISP=SHR,DSN=BETAnn.DB.DEF                       |
|//*                                                             |
|//BQLPRINT DD SYSOUT=*                                          |
|//SYSUDUMP DD SYSOUT=*                                          |
|//BQLIN    DD  *                                                |
|  LOCK DATABASE                                                 |
|/*                                                              |
|//*------------------------------------------------------------ |
|//*       STEP2: BACKUP THE DATABASES WITH ADRDSSU            * |
|//*------------------------------------------------------------ |
|//*                                                             |
|//STEP2   EXEC PGM=ADRDSSU,REGION=0M,COND=(4,LT,STEP1)          |
|//OUTDSN    DD DSN=BACKUP.BETAnn,DISP=(,CATLG),                 |
|//             UNIT=ROBO,LABEL=(1,SL)                           |
|//SYSPRINT  DD SYSOUT=*                                         |
|//SYSIN    DD  *                                                |
| DUMP DATASET (INCLUDE(BETAnn.**) -                             |
| BY(DSORG,EQ,VSAM)) -                                           |
|      SHARE -                                                   |
|      CONCURRENT -                                              |
|      OUTDD(OUTDSN) -                                           |
|      TOLERATE(ENQFAILURE) -                                    |
|      ALLDATA(*) -                                              |
|      OPTIMIZE(4)                                               |
|/*                                                              |
+----------------------------------------------------------------+
```

You can also find a sample job for BST05LAU in the BSA.SAMPLIB in member BQLLSAVE.

Sample unlock job

```
+----------------------------------------------------------------+
|jobcard                                                         |
|//*-------------------------------------------------------------|
|//*       SAMPLE JOB TO RELEASE THE DATABASES                  *|
|//*-------------------------------------------------------------|
|//*                                                             |
|//STEP1   EXEC PGM=BST01RFF,REGION=0M,                          |
|//            PARM='S=nn,BnnLST=00,PGM=BST05CMD,SIGNON=YES'     |
|//STEPLIB  DD  DISP=SHR,DSN=BSA.LOAD                            |
|//         DD  DISP=SHR,DSN=BETAnn.LOAD                         |
|//SFFPARM  DD  DISP=SHR,DSN=BETA.PARMLIB                        |
|//BnnDEF   DD  DISP=SHR,DSN=BETAnn.DB.DEF                       |
|//*                                                             |
|//BQLPRINT DD SYSOUT=*                                          |
|//SYSUDUMP DD SYSOUT=*                                          |
|//BQLIN    DD  *                                                |
|  UNLOCK DATABASE                                               |
|/*                                                              |
+----------------------------------------------------------------+
```

You can also find a sample job for BST05LAU in the BSA.SAMPLIB in member BQLULOCK.

Base Archive Facility (BAF)

Introduction

Overview

The Base Archive Facility (BAF) provides a set of services for writing to archives, reading from archives, and administering archived data. This section describes the LST parameters associated with BAF.

Note: Always see the relevant Beta product documentation for the use of the BAF components and their parameters.

LST parameters for BAF

Keyword	Value	Description	Option	Default
B08_ARCHIVE_SYNC	YES | NO	YES The Beta *nn* started task enqueues multiple requests for the same archive volume internally. If the same volume is requested again, the message 'Waiting for Volume' is received. NO The initiation of the enqueue is handled by the operating system. The message for 'Wait', 'Cancel' or 'Retry' appears when there are multiple requests for the same archive volume.	optional	YES
B08_ARCHIVE_TAPE	*number*	Determines the maximum number of parallel tape units that can be allocated simultaneously for archive view and archive print. Specify B08_ARCHIVE_TAPE=0 if you want to prevent users from using archive view/print for archive tapes.	optional	3
B08_ARCHIVE_TRACE	YES | NO	Specify YES to log all archive and reload commands for Beta *nn*	optional	NO
B08_CONNECT_ACC_ID	*account ID*	These four LST parameters must be coded to archive to the FileTek Storage Machine: B08_CONNECT_ACC_ID B08_CONNECT_PASSWORD B08_CONNECT_SM_NAME B08_CONNECT_SS_ID B08_CONNECT_ACC_ID specifies the account ID of the FileTek Storage Machine.	optional	None
B08_CONNECT_PASSWORD	*Password*	The password of the FileTek Storage Machine.	optional	None
B08_CONNECT_SM_NAME	*name*	The system name of the FileTek Storage Machine.	optional	None
B08_CONNECT_SS_ID	*ssid*	The subsystem ID of the FileTek Storage Machine.	optional	None

Keyword	Value	Description	Option	Default
B08_RELOAD_CATLG	YES | NO	Determines which volume and unit specifications are used for the allocation of archive datasets YES The specifications of the MVS catalog are used. NO The specifications stored in the product database are used (ADR table).	optional	YES
B08_RELOAD_FROM	PRIM | SEC	Use this keyword to override the reload order specified in the archive pool definition: PRIM Reload from primary archive SEC Reload from secondary archive If this keyword is not present, lists are reloaded from the primary or secondary archive according to the archive pool definition.	optional	
B08_RELOAD_SEQ	YES | NO	YES causes archive datasets on tape to be read sequentially during reloading. Code this value if reloading from an archive volume (V3 or higher) has resulted in an error because archive datasets have been reblocked by an IEBGENER compatible utility. NO is the default. Use when reloading data from original volumes written by the archive utility or from copies created by a utility that copies block by block.	optional	NO
OBJ_RETRIEVAL_ DEVICES	*number*	The number of units that can be used simultaneously for retrieving information.	optional	3
OBJ_RELOAD_ CATALOG_*unit*	YES | NO	YES means that the system catalog is used to locate archive datasets during reloading. To prevent the system catalog from being reused for reloading from specific units, specify NO. *unit* refers to the corresponding unit name as defined in the archive subpool.	optional	YES
OBJ_RETRIEVAL_ TIMEOUT	*nn*	Amount of time (in seconds) that a tape or optical disk remains mounted after access	optional	60
OBJ_RETRIEVAL_ ORDER	*n*	Lowest value for the reload order	optional	1

Keyword	Value	Description	Option	Default
OBJ_TRACE_ COMMAND	YES | NO	Activates or deactivates the trace function for commands that are sent to the object server	optional	NO
OBJ_TRACE_ RESERVE	YES | NO	Activates or deactivates the trace function for Unit RESERVE commands that are sent to the object server	optional	NO
OBJ_TRACE_ALLOC	YES | NO	Activates or deactivates the trace function for commands that are sent to the object server	optional	NO

Base Output Facility (BOF)

In this chapter

Topic **Page**

Introduction 280
DD statements in the BOF started task procedure 281
Running BOF as a separate started task 282
BOF default LST parameters for VTAM/APPC 284

Introduction

Overview

The Base Output Facility (BOF) provides a set of services for managing printing requests to 3270-type printers (or emulations). SNA LU types 0, 1, 3 and 6.2 protocols are supported. BOF supports printing to JES and printing to file. Print requests from Beta 93 to VTAM printers are handled by the BOF subsystem, which has to be set up for this purpose. The default name for the BOF started task is BETA07.

Note

Please see the relevant Beta product documentation for the use of the BOF components and their parameters.

What BOF does

The BOF (Beta 07) subsystem does the following:

- It prints to 3270-type VTAM printers.
- It controls the print request queues associated with these printers.

The Beta 07 started task must be active whenever a print request is issued to a VTAM printer by Beta *nn*.

Default names

The following shows the default names generated by the installation procedure:

```
+-----------------+          +------------------+
|  Beta nn        |          |  BOF             |
|                 |          |                  |
|  SSID = BnnA    |--------->|  APPLID = BOF07  |
|  STC  = BETAnn  |          |  SSID = B07A     |
|  B07_SSID=B07A  |          |  STC = BETA07    |
+-----------------+          +------------------+
```

The Base Output Facility subsystem is indicated to Beta *nn* by the keyword B07_SSID in the parmlib member B*nn*LST*xx*.

DD statements in the BOF started task procedure

Overview

This section shows all the DD statements in the BOF started task procedure created during installation.

A minimum region size of 5120K is required.

```
+----------------------------------------------------------------------+
|//BETA07   PROC                                                       |
|//******************************************************************** |
|//*                                                                 * |
|//*  BETA07 (BOF) SEPARATE STARTED TASK VERSION 7                   * |
|//*  SAMPLE-PROCEDURE                                               * |
|//*                                                                 * |
|//*                                                                 * |
|//******************************************************************** |
|//BETA07   EXEC PGM=BST01SFF,TIME=1440,REGION=5120K,                  |
|//             PARM='S=07,B07LST=00,B01LST=00'              <======   |
|//STEPLIB  DD  DSN=BSA.LOAD,DISP=SHR                       <======   |
|//SFFPARM  DD  DISP=SHR,                                             |
|//             DSN=BETA.PARMLIB                             <======   |
|//B07DEF   DD  DUMMY                                                 |
|//IMAGELIB DD  DISP=SHR,DSN=SYS1.IMAGELIB                            |
|//*                                                                  |
|//SFFFDUMP DD  SYSOUT=7,DCB=BLKSIZE=6050,FREE=CLOSE                  |
|//SFFFDUMP DD  SYSOUT=7,DCB=BLKSIZE=6050,FREE=CLOSE                  |
|//SFFFDUMP DD  SYSOUT=7,DCB=BLKSIZE=6050,FREE=CLOSE                  |
|//SFFFDUMP DD  SYSOUT=7,DCB=BLKSIZE=6050,FREE=CLOSE                  |
|//*                                                                  |
|//SYSABEND DD  SYSOUT=7                                              |
|//SNAPDUMP DD  SYSOUT=7                                              |
+----------------------------------------------------------------------+
```

BOF STC proc DD statements

DD name	Description
STEPLIB	BSA load library
SFFPARM	Beta parameter library
B07DEF	Dummy database definition file
SFFFDUMP	Datasets for BOF started task dumps
IMAGELIB	Dataset for VTAM printer FCB support

Running BOF as a separate started task

Overview

The Base Output Facility (BOF) can be run as a separate started task. In this case you will need to specify the subsystem ID and initialize the BOF subsystem. An optional entry can be made in member IEFSSN*xx* in your system IPL is necessary to activate the changes in parmlib member IEFSSN*xx*.

Procedure

Code one line in member IEFSSN*xx*, starting in column 1. Columns 1 to 4 contain the subsystem ID, optionally followed by a program name and a parameter. The line in member IEFSSN*xx* should look like the following:

```
B07A,BST01ARI,'BETA.PARMLIB(B07SSI00)'
```

Where B07A is the Beta 07 subsystem ID you specified in the installation procedure, and BST01ARI is the program to be executed. This program is executed at IPL time. It initializes the Beta 07 security environment.

Note: BST01ARI must reside in an APF-authorized library concatenated in the linklist, link-edited with AC(1).

Parameters for BST01ARI

Member B07SSI00 of the BETA.PARMLIB contains a set of parameters for the program BST01ARI:

UXSRT=BST00STH	Specifies the module that calls the product security exit (module name must be BST00STH).
UXSIN=IEFBR14	Specifies the BOF security exit module (default name IEFBR14).
SVC=	The Beta SVC number.

BST01ARI loads the modules specified in the parameter member into the CSA, subpool 241. The specified modules have to reside in an APF-authorized library concatenated in the linklist. The default name of this library is BETA.APFLOAD.

Running BST01ARI

The program BST01ARI must always run before the Beta 07 subsystem can be started the first time. This is usually done at IPL time. However, BST01ARI can also run as a batch job. This is useful, for example, if the security exit has been modified, but you do not want to perform an IPL to activate the changes.

Sample JCL for BST01ARI

Member B07INIT in the BETA93.CNTL provides sample JCL for BST01ARI. The program parameters are passed as follows:

```
INIT EXEC PGM=BST01ARI,PARM='B07A,BETA.PARMLIB(B07SSI00)'
```

where B07A is the subsystem ID, and BETA.PARMLIB(B07SSI00) is the member containing the names of the modules to be loaded (see above).

Security

To prevent unauthorized access to the program BST01ARI, a RACROUTE is issued when a user submits this batch job. The RACF request is

```
REQUEST=AUTH,CLASS='FACILITY',ACCESS=READ
```

The entity name is ***B07.INIT.ssid***, where *ssid* is the subsystem ID coded in the EXEC parameter.

BST01ARI return codes

0 Program terminated normally.

4 Program terminated with a warning for one of the following reasons:

- The profile 'BETA.INIT.*ssid*' not defined.
- Specified module could not be loaded because it was not found, or because it was not in an authorized library.
- FREEMAIN failed.

8 Program terminated abnormally due to parse error. Statements in member IEFSSNxx were not correctly coded.

12 Program terminated abnormally because GETMAIN failed.

16 Program terminated abnormally for one of the following reasons:

- ENQUEUE failed (e.g. started task was active)
- DYNALLOC or OPEN error (e.g. parameter dataset or member not found)
- ESTAE failed (recovery)
- BST01ARI not authorized

20 Task abended.

BOF default LST parameters for VTAM/APPC

VTAM printer support/ APPC connection

The Base Output Facility subsystem has its own parmlib member with the name B07LST*xx*. This section describes all the keywords used by the Base Output Facility started task. Please see the Beta *nn* product documentation for further details.

Keyword	Parameter	Description	Option	Default
B07_SSID	Subsystem ID	The subsystem ID used to identify the started task.	Required	B07A
B07_INFOMSG	YES / NO	Specifies whether informational messages are displayed.	Required	YES
B07_COMMENT	Message	The start up message displayed on the console when the started task has completed start up processing. The startup message has the number 9190I. The default text is: "BETA BASE OUTPUT FACILITY IS NOW ACTIVE"	Required	See description
BQL_MASTER_ SSID	Comma	This keyword has to be present in every B93LST*xx* member. The parameter must be a comma.	Required	Comma
B07_APPLID	VTAM-Appl-ID	The name of the VTAM application. This name must be defined to VTAM by means of an APPL statement in your installation's VTAMLST dataset.	Optional	B07VTAM
B07_BRACKET_ PERM	YES / NO	Sends a VTAM-BID request to the printer to obtain permission to use the bracket protocol.	Optional	NO
B07_RESPONSE_ TOT	0 – 60	The max. response time (in minutes) allowed for a printer logon request. If the maximum time is reached without a successful printer logon, the request is aborted with the sense code 55090001. **0** means that the print logon request continues waiting. The wait-time is only valid for abnormal waiting situations and not, for example, for the pre-allocation (PREALC) status.	Optional	10

Keyword	**Parameter**	**Description**	**Option**	**Default**
B07_CHAIN_LU1	0-8	This keyword is valid for all LU1 printers. The value represents the size of the chain buffers in Kilobytes. The maximum size allowed is 8K. A reasonable and sufficient working size is 4K. **0** means that no buffer chaining is used. Example: B07_CHAIN_LU1 = 4.	Optional	0
B07_CHAIN_LU3	YES / NO	This keyword is valid for all LU3 printers (NO stands for inactive Chaining). The chaining buffer size is calculated from the Printer Logmode Definition (screen-/alternate screen-size).	Optional	NO
B07_BINARY_ SUPPORT	YES / NO	Supports the processing of ASCII-files/data	Optional	NO

Diagnostic reports and traces

In this chapter

Topic	Page
Information for Beta Systems support	287
BSA trace parameters	288
BSA trace media	290
Allocating a trace medium	291
Releasing a trace medium	293
Activating/Deactivating BSA traces	294
Diagnostic reports	295
Beta SMP/E package information	298

Information for Beta Systems support

Overview

Activate BSA traces to provide additional information for Beta Systems support. BSA traces are useful when you are trying to analyze specific system activities. The following sections provide information on:

- Which BSA traces are available
- Which media can be used to log BSA trace output
- How to activate and deactivate BSA traces
- How to allocate and free a medium

Tracing BSA components

BSA traces can be logged for different BSA components:

- Beta Query Language (BQL)
- Base Output Facility (BOF)
- Base Archive Facility (BAF)
- BSA TCP/IP Server
- BSA Service Manager (BSM)

Diagnostic reports

BSA provides a diagnose and support tool that collects information on the runtime environment and writes a diagnostic report.

The writing of diagnostic reports and dumps can be initiated using the following alternative methods:

- LST parameter DIAG= (for batch job or STC)
- Operator console command (for STC only)
- BSA Service Manager option **S.R** (for STC only)

For more information, see "Diagnostic reports" on page 295 and the *BSA Service Manager Manual*.

BSA trace parameters

Overview

All BSA trace parameters are optional. They are not generated automatically during installation.

All parameters that can be updated dynamically are displayed under the BSA Service Manager option **1.2**.

Allowed values are **NO** and **YES** for most parameters, except the following:

BSA_TCPIP_TRACE = NO **|** YES **|** SHORT (or SHO) **|** ERROR (or ERR)

BSA_TRACE_SEC = NO | YES | ALL

Parameters

Parameter name	**Description**	**Component**	**Default**
BSA_LICENSE_TRACE	Logs all license check activities	Base component	NO
BSA_TCPIP_TRACE	Logs all program activities of TCP/IP applications Instead of YES, specify one of the following values if you want to reduce the amount of information being logged: • SHORT (or SHO) • ERROR (or ERR)	TCP/IP server	NO
BSA_TCPIP_TRACE_ INTERN	Logs all TCP/IP requests.	TCP/IP server, Base Output Facility (BOF)	NO
BSA_TCPIP_TRACE_ SNDRCV	Logs all TCP/IP SEND or RECEIVE requests This will only come into effect if keyword TCPIP_TRACE_INTERN has been activated beforehand. If NO, only the first SEND or RECEIVE request will be logged.	TCP/IP server, Base Output Facility (BOF)	YES
BSA_TCPIP_TRACE_BUF	Logs all buffer data of SEND or RECEIVE requests in dump format	TCP/IP server, Base Output Facility (BOF)	NO
BSA_TCPIP_TRACE_LST	Logs all the LST keywords that are processed while the server is starting	TCP/IP server	NO
BSA_TCPIP_TRACE_OCF	Logs the establishing of connections when using OCF via TCP/IP	TCP/IP server	NO

Parameter name	**Description**	**Component**	**Default**
BSA_TRACE	Logs all BSA activities	All components	NO
BSA_TRACE_SEC	Logs all RACF checks (including the WTO messages with the security profile if the product exit B*nn*UXSEC supports this) Instead of YES, specify ALL if you also want to log all RACINITs (LGON/LGOFF) in addition to the RACF checks.	Base System Facilities (BSF)	NO
BQL_TRACE	Logs all BQL and BOF commands	Beta Query Language (BQL), Base Output Facility (BOF)	NO
B08_ARCHIVE_TRACE	Logs all archive/reload commands	Base Archive Facility (BAF)	NO
B07_TCPIP_TRACE	Traces all commands used for TCP/IP requests in BOF	Base Output Facility (BOF)	NO
B07_VTAM_TRACE	Logs all information that comes up when an initialization of a VTAM print connection is attempted	Base Output Facility (BOF)	NO
OBJ_OBJECT_TRACE	Logs all reload and retrieval commands (_beta doc|z plus)	Base Archive Facility (BAF)	NO
OBJ_TRACE_RESERVE	Activates or deactivates the trace function for unit reserve commands that are sent to the object server	Base Archive Facility (BAF)	NO
OBJ_TRACE_ALLOC	Traces commands that are sent to the object server	Base Archive Facility (BAF)	NO

BSA trace media

BSA trace media

The following output media can be allocated instead of the default SYSLOG and JESMSGLG:

SYSOUT	An output class **Note:** We recommend that you use an output class (SYSOUT) as the medium for the trace output.
SUBSYS	A subsystem
A dataset	An existing sequential dataset This dataset must be allocated with disposition SHR. The started task or batch job must have WRITE access to this dataset. **Note:** If the dataset becomes full, the system will continue to log data in this dataset, starting from the beginning. This means that data that has been stored up to this point will be lost.

Allocating a trace medium

Overview

The default writes trace output to the system log (SYSLOG) and the job log (JESMSGLG) of a started task (STC) or batch job.

First, allocate a medium for the BSA trace. Next, activate the BSA trace via BSA trace keywords. See details below.

Allocation methods

Two methods are available to allocate a medium. The method chosen depends on whether a started task or batch job will be used:

- dynamic medium allocation for a started task (STC). This uses the MODIFY command. Note: Use dynamic allocation for started tasks only.
- direct medium allocation for a batch job by means of an entry in the DD statement BSATRACE.

Dynamic allocation

To dynamically allocate a medium for a started task (STC), you can either use online option **2.7** of the BSA Service Manager for the Dynamic Trace Facility, or use the following MODIFY command:

```
F stcname,TR ALLOC 'output medium'
```

Enter the name of the started task in use and replace medium by the medium you want to allocate – an output class, a subsystem or a dataset (see "BSA trace media" on page 290). You can enter a total of 128 characters for the entire MODIFY command.

Replace...	**with ...**	**Details**	**Parameters**
started task name	the started task (STC)	the name of the started task in use	-
'output medium'	an output class	SYSOUT(c[,*ext,form*])	Replace c with the output class. In addition, you can enter the extension and the format.
	a subsystem	SUBSYS(s[,*form,ext*])	Replace s with the subsystem ID. In addition, you can enter the format and the extension.
	a dataset	the name of the dataset you want to use (if you want to suppress trace output, enter **dummy** or **nullfile**)	-

Examples

```
F stcname,TR ALLOC 'SYSOUT(P,TRACE,STD)'
F stcname,TR ALLOC 'SUBSYS(B93P,STD,TRACE)'
F stcname,TR ALLOC 'BSA.TEST.TRACE'
```

Result for the STC

Once the output medium has been successfully allocated (message 9152I), the requested trace data and all product messages will be logged on the specified medium from this point on. The trace output is displayed under the DD statement BSATRACE.

Direct allocation

To allocate a medium for a batch job, specify the medium directly in the DD statement BSATRACE.

Examples

```
//BSATRACE  DD  SYSOUT=(P,TRACE,STD)
//BSATRACE  DD  SUBSYS=(B93P,STD,TRACE)
//BSATRACE  DD  DSN=BSA.TEST.TRACE,DISP=SHR
```

Result for batch

Once the output medium has been successfully allocated (message 9152I), the requested trace data and all the product messages will be logged onto the specified medium from this point on. The trace output is displayed under the DD statement BSATRACE.

Releasing a trace medium

Medium used for an STC

To dynamically free a medium used for a started task (STC), use online option **2.7** of the BSA Service Manager for the Dynamic Trace Facility, or use the following MODIFY command:

```
F stcname,TR FREE
```

Replace *stcname* with the name of the STC concerned. It is not necessary to define medium name itself.

Once the medium has been successfully released, you will receive message 9165**I**.

Note

Provided that the BSA traces have not been deactivated, trace output will be stored in the system log (SYSLOG) and in the JESMSGLG of the started task (STC) or batch job as before.

Medium used for a batch job

It is not necessary to free a medium used for a batch job. Simply stop the job concerned and delete the DD statement BSATRACE before restarting it.

Activating/Deactivating BSA traces

Overview

Some BSA traces can be dynamically activated and deactivated while the started task (STC) is running. Others can only be activated and deactivated via the corresponding LST parameter in the EXEC parameter or parmlib member of the Beta Systems product. To find out whether a trace can be activated/deactivated dynamically, see column **DAA?** (**D**ynamic **A**ctivation **A**llowed) in the table of trace keywords (see "BSA trace parameters" on page 288).

Dynamic activation/ deactivation

You can activate/deactivate BSA traces dynamically with the help of the BSA Service Manager. You can use the "Display Modifiable Parameter Keywords/Values" panel (option **1.2**) or you can use the "Dynamic Trace Activation/Deactivation" panel (option **2.7**).

To activate/deactivate a BSA trace via the "Display Modifiable Parameter Keywords/Values" panel:

1. In the "Primary Selection Menu" of the Beta product, enter **D.S** to display the BSA Service Manager.
2. Select option **1.2** to go to the "Display Modifiable Parameter Keywords/Values" panel.
3. Enter line command **U** (update) or **I** (insert) in front of the corresponding BSA trace keyword, enter the value **Yes** to activate or **No** to deactivate the trace, and then enter **Y** to confirm your change.

Direct activation/ deactivation

You can activate/deactivate BSA traces directly by specifying the corresponding LST keyword and value in the EXEC parameter or parmlib member of the Beta Systems product. (Parmlib members B*nn*LST*xx*, where *nn* represents a Beta product number and *xx* any combination of numeric values or characters, are always used when the started task (STC) or batch job is started.)

To activate/deactivate a BSA trace directly via the corresponding LST parameter:

1. Open the corresponding member in the BETA.PARMLIB, e.g. B93LST00.
2. Specify the trace keyword and value, e.g. BQL_TRACE=YES or BQL_TRACE=NO.
3. Stop and restart the started task (STC).
4. In case of activation, allocate the medium as described (see "Allocating a trace medium" on page 291).

Diagnostic reports

Overview

The STC or batch job can write a diagnostic report, which contains system-related environment information on the started task or batch job. The diagnostic report is written to DD BSADIAG, which is allocated dynamically as necessary. The writing of BSADIAG can be initiated using the following alternative methods:

- Service Manager option **R.*n***
- Operator console command
- LST parameter DIAG=

Service Manager options

You can use the BSA Service Manager online options to create a diagnostic report (DD BSADIAG):

- Use option **R.1** to obtain BSADIAG via a separate batch job
- Use option **R.2** if the STC itself is to allocate BSADIAG

The fields of the panels displayed under Service Manager options **R.1** and **R.2** correspond to the parameters of the DIAG parameter and MODIFY command described below. For more information, see *BSA Service Manager Manual*.

Operator console command

The following MODIFY command initiates the writing of a diagnostic report to DD BSADIAG:

```
F stcname,DIAG[=ocl[#FREE|HOLD[#dcl[#DUMP|FDMP|NDMP[#chd]]]]]
```

where:

stcname	is the name of the started task.
ocl	is the output class to be used for DD BSADIAG. If * is coded, the message class of the batch job or STC is used.
FREE|HOLD	Specify FREE to control whether BSADIAG is to be released immediately, or HOLD to hold output until the batch job or STC terminates or until manual release. HOLD is the default.
dcl	is the output class to be used to the dump. If * is coded, the message class of the batch job or STC is used.
DUMP|FDMP|NDMP	Specify one of the following: DUMP causes the writing of a diagnostic dump; BSADIAGA/BSADIAGD remains allocated until manual release or termination of the batch job or STC FDMP causes the writing of a diagnostic dump; BSADIAGA/BSADIAGD is released after writing NDMP no dump
chd	is the problem number (if available) for which you are providing information. This number will be included in the diagnostic report.

# is used as separator between parameters. All parameters are optional, but if a parameter is coded, values for all preceding parameters must be coded as well.

Examples

```
F stcname,DIAG
F stcname,DIAG=*#HOLD
F stcname,DIAG=*#HOLD#*#NDMP#123-R123-1234
```

LST parameter DIAG=

The LST parameter DIAG= can be coded in the LST member or in the EXEC parameter of the JCL (recommended). It is mainly intended for use with batch jobs, but it can also be used for STCs as an alternative to the console command or the corresponding Service Manager options. The syntax of the LST parameter DIAG corresponds to the syntax of the MODIFY command:

DIAG=*ocl*[#FREE|HOLD[#*dcl*[#DUMP|FDMP|NDMP[#*chd*]]]]

Example

Following is an example where the LST parameter DIAG has been coded in the EXEC parameter of the Beta 93 archive batch utility. The diagnostic report is written after all function groups have been loaded. DD BSADIAG is allocated dynamically using the message class of the batch job.

```
+-----------------------------------------------------------------+
|jobcard                                                          |
|//B93DEARC EXEC PGM=BST01RFF,REGION=0M,PARM=('S=93',             |
|//             'PGM=B93ARC',                                     |
|//             'B01LST=00',                                      |
|//             'B93LST=00',                                      |
|//             'DIAG=*#HOLD#*#NDMP#123-R123-1234',               |
|//             'SIGNON=YES')                                     |
|//...                                                            |
+-----------------------------------------------------------------+
```

BSADIAG

The diagnostic report written to DD BSADIAG contains the following (amount of information varies depending on the object of the diagnostic report):

- SFF SVC/EXIT information
- LST parameters used
- System symbol table
- Subsystem interface control area (SICA)
- Subsystem control table (SSCT)
- Subsystem control area (SSCA)
- Subsystem routine vector table (SSVT)
- Overview of all Beta subsystems defined to z/OS (including information on OCF and XCF connections)
- SFF transaction list
- Program module information (Steplib and Linklist)
- SMP/E package information (status of all packages; if status is MISMATCH or CHANGED, also package contents (load modules only))

Beta SMP/E package information

Overview

The STC or batch job can output information on Beta SMP/E packages. The writing of this information can be initiated using the following alternative methods:

- Service Manager option **R.P**
- Operator console command
- LST parameter BSA_ANALYZE_SMPE = YES

Service Manager option

You can use the BSA Service Manager online option **R.P** to display information on Beta SMP/E packages.

For more information, see *BSA Service Manager Manual*.

Operator console command

The following MODIFY command initiates the verification of Beta SMP/E packages:

```
F stcname,DSMPE [pkgname] [LNK]
```

where:

stcname	is the name of the started task.
pkgname	is the name of an SMP/E package (user package or consolidated service package) or a mask Specify a fully qualified name to receive information on the load modules contained in the specified package, including the status of each load module. Specify a mask (or leave blank for all packages) to receive information on the matching packages, including the status of each matching package. You can use **?** (= any single character) in any position and ***** (= any sequence of characters) in trailing position. For example: **U??61*** selects all user packages of V6R1 **X??61*** selects all service packages of V6R1
LNK	If specified, the linklist and LPA libraries will be scanned for SMP/E packages in addition to the steplib concatenation.

Example 1

```
F stcname,DSMPE
```

```
PMS9155I OPERATOR MODIFY COMMAND 'DSMPE' RECEIVED
PMS9350I SMP/E PACKAGE INFORMATION
PMS9350I PKGNAME  PKGMEM   PKGPTF   STATUS     DATE
PMS9350I XBS61L02 BST61L02 PBS0296  LMOD OK    13.08.2015
PMS9350I UBS61205 BST61205 PBS0248  OK         04.06.2015
PMS9350I UBS61206 BST61206 PBS0252  CHANGED    04.06.2015
PMS9350I UBS61207 BST61207 PBS0276  LMOD OK    27.07.2015
PMS9350I XPM61L02 B9361L02 PPM6410  CHANGED    27.08.2015
PMS9350I UPM61109 B9361109 PPM6315  CHANGED    02.06.2015
PMS9350I UPM61110 B9361110 PPM6336  CHANGED    02.06.2015
PMS9350I UPM61111 B9361111 PPM6403  OK         13.08.2015
PMS9350I UPM61112 B9361112 PPM6395  CHANGED    21.07.2015
PMS9350I ****** BOTTOM OF SMP/E PACKAGE INFORMATION *****
```

Example 2

```
F stcname,DSMPE XBS61L02
```

```
PMS9155I OPERATOR MODIFY COMMAND 'DSMPE XBS61L02' RECEIVED
PMS9351I SMP/E PACKAGE XBS61L02 DETAIL INFORMATION
PMS9351I PKGMEM   PKGPTF   CURRPTF  STATUS     CURRDATE   CURRTIME
PMS9351I BST01SSI PBS0004  PBS0004  LMOD OK    28.02.2014 16.28
PMS9351I          STEPLIB  SHRCOM.L146102.APFLOAD
PMS9351I BST09SPR PBS0017  PBS0017  LMOD OK    25.03.2014 12.06
PMS9351I          STEPLIB  SHRCOM.L146102.LOAD
PMS9351I BST02SND PBS0020  PBS0020  LMOD OK    25.04.2014 08.16
PMS9351I          STEPLIB  SHRCOM.L146102.LOAD
PMS9351I BST09OU3 PBS0021  PBS0021  LMOD OK    25.04.2014 12.04
PMS9351I          STEPLIB  SHRCOM.L146102.LOAD
PMS9351I BST01SV  PBS0057  PBS0057  LMOD OK    08.05.2014 15.59
PMS9351I          STEPLIB  SHRCOM.L146102.LOAD
...
PMS9351I ****** BOTTOM OF SMP/E DETAIL PACKAGE INFORMATION *******
```

LST parameter BSA_ANALYZE_SMPE=

The LST parameter BSA_ANALYZE_SMPE = YES can be coded in the B*nn*LST*xx* member of the product or in the EXEC parameter of the JCL if you always want to verify the Beta SMP/E packages (message 9350I) when starting the STC or batch utilities. The default of this parameter is NO.

Third-party software

Overview

This section provides a list of the third-party software that is included in Beta Systems Architecture together with the license terms under which this third-party software is distributed.

License texts

The license texts are available in the software download in the directory **Third Party Licenses**.

Copyright

Third-party software is the copyright of its respective copyright owner as indicated in the corresponding text in **Third Party Licenses**.

Disclaimer

Beta Systems DCI Software AG assumes no liability with regard to the use of third-party software and makes no representations or warranties as to the usability, functionality, accuracy or freedom from error of third-party software.

Software list

Clarified Artistic License

- NcFTPPut

  Version 3.1.9

  https://www.ncftp.com/ncftp/doc/LICENSE.txt

MIT License

- PuTTY

  Version 0.60

  https://github.com/github/putty/blob/0.60/LICENCE

zlib License

- zlib

  Version 1.2.3

  http://www.zlib.net/zlib_license.html

Index

Symbols & Numeric

- $BETA 82

A

- activate authorized functions 49
- address space for XCF 131
- APFLOAD 30, 48
- APPC services 89
- application check 142
- application ports 139
- ARM 214
- AT-TLS 163
  - Policy Agent 164
  - policy rules 169
- authentication security 192
- authorized functions 48
- automatic database extension 224
- automatic restart management 214

B

- B01LSTxx keywords 69
- B02UXSIN 142
- backing up a running database 272
- Base Archive Facility (BAF) 11, 275
- base function facilities 11
- Base Output Facility (BOF) 11, 279
- batch jobs for installation 42
- Beta 02 107
  - Beta 02 as an independent STC 144
- Beta SMP/E packages 298
- Beta SVC *See* SVC
- BETA.PARMLIB 60, 63
  - syntax rules 66
- BETAGXCF 134
- BnCLSYN 237
- BnnLSTxx 63
- BnnLSTxx keywords 71
- BnnSSIxx 51, 56, 134
- BOF *See* Base Output Facility (BOF)
- BQL database facilities 11
  - BQL SHARE 232
  - BSA database batch utilities 242
  - building keys for a table 260
  - database definition file 246
  - deleting contents 261
  - formatting a database 258
  - global LST parameters for BQL 221
  - loading a database 255
  - locking and unlocking 254
  - LST parameters for BQL SHARE 234
  - serialization 235
  - synchronization 237
- BQL_OPEN_FILES 245
- BSA Communication Integrator 174
  - JCL for the BSA Communication Integrator 184
  - LST parameters for BSA CI 204
- BSA components overview 11
- BSA control block repair 262
- BSA cross memory 118
- BSA database batch utilities 242
- BSA environment 40
- BSA libraries 9
- BSA XCF Global Connect 130
- BSF 11
- BSM - BSA service manager 11
- BST01ARI 51, 56, 134
  - initializing subsystems for XCF 120
- BST01MST 49
- BST05ANA 262
- BST05CMD 245
- BST05DBL 255, 257, 260, 261
- BST05FOR 258
- BST05LAU 254

BST05SYC 239
BST05UPF 246
BST08OCP 270
BSTUXASM 142
BxxUXSEC 84

C

CAF - CICS access facility 11
CDT 82
certification 192
checking archive files 270
checklist for installation 30
class $BETA 81
class descriptor table 82
client identification 141
Communication Integrator 174
communication outside the sysplex 122
compression and encryption 141, 210
concatenated VSAM datasets (EXT01, EXT02 etc.) 228
concurrent copy 272
copying archive files 270
cross coupling facility 11

D

database batch utilities 242
database definition file 246
database extension, dynamic 224
database sharing 232
DD BQLIN 247, 256, 259, 260, 261
digital key ring 200
DMF database facilities 11
dynamic classes (CDT) 82
dynamic database extension 224
dynamic server control 138

E

EDF 147
enclaved SRB 212
encryption and compression 141, 210
environment 9, 40
EXEC statement 63
exits 100
  logon exit (B02UXSIN) 142
  security exit 84
extended VSAM datasets 224
EXTnn (concatenated VSAM datasets) 228

F

formatting a database 258

G

Global Connect for XCF 130
global LST parameters for BQL 221
global ports 139
global resource serialization 236
GRS 236

H

hardware requirements 29, 30
high water mark 224
host-to-host security 100

I

I#BSAJ01 43
I#BSAJ02 44
I#BSAJ03 44
I#BSAJ04 44
I#BSAJ05 44
I#BSAJ06 44
I#BSAJ07 45
I#BSAJ08 46
I#BSAJ09 46
I#BSAJ10 47
I#BSAJ11 47
I#BSAJ12 47
I#BSAJ13 47
IAF - IMS access facility 11
IBM licensed products 29
IEESYSAS 134
index file definitions OFFLINE/ONLINE 245
initialization 51
  initializing subsystems for XCF 120
installation batch jobs 42
installation checklist 30
installation ID 22
  installation ID unspecified 25
installation REXX 40
installation steps 34

IPL 49
  defining/initializing subsystem 52
  specifying/activating Beta SVC 48
IXCLEAVE 131

K

key ring 200
keywords for traces 288

L

licenses 18
  definition of terms 25
  installation ID 22
  license check 22
  license file parameters 27
  license file pointer 19
  license file structure 18
licensing 18
logon exit (B02UXSIN) 142
LST members 63
LST parameters 69
  B01LSTxx keywords 69
  BnnLSTxx keywords 71
  global LST parameters for BQL 221
  keywords for the TCP/IP server 157
  LST parameters for BAF 276
  LST parameters for BOF 284
  LST parameters for BQL SHARE 234
  LST parameters for BSA CI 204
  OCF LST keywords 95

M

MAXDATA 98
message suppression 76
MIPS license type 19
model spool file definitions 241
multi-CPU with OCF 86
  multi-CPU security 100
  network resource definitions 94
  OCF configuration 91, 102
  OCF environment 89
  OCF functions and parameters 88
  OCF LST keywords 95
  VTAM resource definitions 93
multi-CPU with XCF 116
  communication outside the sysplex 122
  initializing subsystems for XCF 120
  XCF communication 119, 122
  XCF configuration 118
  XCF configuration examples 123, 126

N

naming conventions 30
network resource definitions 94
nodes (OCF) 89

O

OCF 11
  multi-CPU with OCF 86
  network resource definitions 94
  OCF configuration 91, 102
  OCF environment 89
  OCF functions and parameters 88
  OCF LST keywords 95
  operator console commands for OCF 101
  SFF services 89
  VTAM resource definitions 93
OFFLINE/ONLINE spool and index files 245
OMVS segments 19
open communication facility 11
  multi-CPU with OCF 86
  network resource definitions 94
  OCF configuration 91, 102
  OCF functions and parameters 88
  OCF LST keywords 95
  operator console commands for OCF 101
  SFF services 89
  VTAM resource definitions 93
operating system 29
operator console commands for OCF 101

P

PAGENT 164
parmlib 60, 63
  syntax rules 66
passwords
  mixed-case passwords 142
  passphrase support 142

Policy Agent 164
policy rules 169
PORT LST parameter (CI) 204
PORT LST parameter (TCP/IP server) 154

R

RACF definitions 82
RACF security user table 85
recovering spool files 262
REFRESH 85
reorganizing spool files 262
repairing archive files 270
report generator 11
request unit size 98
restoring archive files 270
REXX 40
RFF 60, 61
  RFF job for EDF 147
  suppressing dumps for SSF/RFF 77
  syntax rules 66
  TCP/IP server 136
routing codes 71
RU sizes 98

S

secondary space 224
secure sockets layer *See* SSL/TLS
security 81
  multi-CPU security 100
security exit 84
serialization 235
  GRS 236
server licensing 19
service packages 298
service ports 186
SFF 11, 60, 61
  SFF services 89
  suppressing dumps for SSF/RFF 77
  syntax rules 66
  TCP/IP server 136
SHARE 232
SMP/E environment 9, 40
SMP/E packages 298
software requirements 29
spool files
  setting OFFLINE/ONLINE 245
  spool file analysis 262
  spool file recovery (SFR) 262
  spool reorganization 262
SSL/TLS 192
  service ports 186
  See also AT-TLS
static classes (CDT) 82
static system symbols 79
subsystem function facility 11
subsystems
  initializing a subsystem 51
  subsystem unavailable message 126
  TLS *See* AT-TLS; SSL/TLS
suppressing dumps for SSF/RFF 77
suppression
  message suppression 76
  suppressing dumps for SSF/RFF 77
SVC 47
  (re)linking 47
  activating 49
  specifying to z/OS 48
  SVCUPDTE 49
SYNC file 237
sysplex 116
  communication outside the sysplex 122
  sysplex operation 124
  XCF communication 119, 122
  XCF configuration 118
  XCF configuration examples 123, 126
SYSVAR support 77

T

TCP/IP ports 139
TCP/IP server 136
  application check 142
  AT-TLS 163
  Beta 02 as an independent STC 144
  compression and encryption 141, 210
  dynamic server control 138

logon exit (B02UXSIN) 142
LST parameters for the BSA TCP/IP Server 154, 157
operator commands 150
ports 139
TCP/IP server in the Beta product 146
third-party software 300
traces 286
keywords for traces 288

U

usage licensing 19
user PTF packages 298

V

VAF - VTAM access facility 11
VDF - VTAM dialog facility 11
VTAM resource definitions 93

W

web applications 136
WLM 212
WLM_SUBSYS_TYPE 213
workload manager 212

X

XCF 11, 116
communication outside the sysplex 122
initializing subsystems for XCF 120
XCF communication 119, 122
XCF configuration 118
XCF configuration examples 123, 126
XCF Global Connect 130
X-system router 107
Beta 02 as an independent STC 144

Z

z9 Integrated Information Processor 211
ZIP (zIIP facility) 211