



Configuration Reference

Version 6.2, June 2005

IONA, IONA Technologies, the IONA logo, Orbix, Orbix/E, Orbacus, Artix, Orchestrator, Mobile Orchestrator, Enterprise Integrator, Adaptive Runtime Technology, Transparent Enterprise Deployment, and Total Business Integration are trademarks or registered trademarks of IONA Technologies PLC and/or its subsidiaries.

Java and J2EE are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

CORBA is a trademark or registered trademark of the Object Management Group, Inc. in the United States and other countries. All other trademarks that appear herein are the property of their respective owners.

While the information in this publication is believed to be accurate, IONA Technologies PLC makes no warranty of any kind to this material including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. IONA Technologies PLC shall not be liable for errors contained herein, or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

COPYRIGHT NOTICE

No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, photocopying, recording or otherwise, without prior written consent of IONA Technologies PLC. No third party intellectual property right liability is assumed with respect to the use of the information contained herein. IONA Technologies PLC assumes no responsibility for errors or omissions contained in this book. This publication and features described herein are subject to change without notice.

Copyright © 2004 IONA Technologies PLC. All rights reserved.

All products or services mentioned in this manual are covered by the trademarks, service marks, or product names as designated by the companies who market those products.

Updated: 04-Jul-2005

Contents

Preface	vii
Chapter 1 Introduction	1
Orbix Configuration Concepts	2
Configuration Data types	4
Chapter 2 Root Namespace	7
Chapter 3 COMet	9
COMet:Config	10
COMet:Mapping	12
COMet:Debug	13
COMet:TypeMan	14
COMet:Services	18
Chapter 4 Core Namespaces	19
initial_references	20
binding	23
buffer	26
domain_plugins	27
event_log	28
orb_management	29
poa:fqpn	30
thread_pool	32
url_resolvers	35
Chapter 5 Classloader	37
classloader	38
Chapter 6 Configuration Namespace	41
configuration	42

Chapter 7 CORBA Plug-ins	43
plugins:atli2_ip	46
plugins:atli2_shm	47
plugins:basic_log	49
plugins:codeset	50
plugins:config_rep	54
plugins:connection_filter	55
plugins:egmiop	56
plugins:event	58
plugins:event_log	62
plugins:giop	63
plugins:giop_snoop	64
plugins:http(s)	67
plugins:i18n	71
plugins:iiop	73
plugins:ifr	78
plugins:it_http_sessions	80
plugins:it_mgmt	81
plugins:it_mbean_monitoring	82
plugins:it_pluggable_http_sessions	83
plugins:it_response_time_collector	85
plugins:it_security_service	87
plugins:file_security_domain	88
plugins:jta	89
plugins:local_log_stream	91
plugins:locator	93
plugins:naming	96
plugins:node_daemon	99
plugins:notify	101
plugins:notify:database	105
plugins:notify_log	109
plugins:orb	110
plugins:ots	111
plugins:ots_lite	116
plugins:ots_encina	118
plugins:ots_mgmt	125
plugins:poa	127
plugins:pss	128
plugins:pss_db:envs:env-name	129

plugins:pss_db:envs:env-name:dbs:storage-home-type-id	141
plugins:shmiop	144
plugins:tlog	145
plugins:tlog:database	148
plugins:ziop	152
Chapter 8 CORBA Policies	153
Core Policies	154
CORBA Timeout Policies	156
Orbix-Specific Timeout Policies	157
policies:ajp	158
policies:binding_establishment	159
policies:egmiop	161
policies:giop	162
policies:giop:interop_policy	164
policies:http(s)	167
policies:iiop	169
policies:invocation_retry	174
policies:shmiop	176
policies:well_known_addressing_policy	177
policies:ziop	178
Chapter 9 JMS	181
destinations	182
factory	183
instrumentation	184
jmx:adaptor	185
persistence	186
plugins:jms	188
Chapter 10 Security	189
Applying Constraints to Certificates	191
initial_references	193
plugins:atli2_tls	194
plugins:csi	195
plugins:gsp	196
plugins:https	201
plugins:iiop_tls	202

CONTENTS

plugins:kdm	208
plugins:kdm_adm	210
plugins:locator	211
plugins:schannel	212
plugins:security	213
policies	214
policies:csi	220
policies:https	223
policies:iiop_tls	228
principal_sponsor	238
principal_sponsor:csi	242
Chapter 11 XA Resource Manager	245
Glossary	247
Index	255

Preface

Orbix is a software environment for building and integrating distributed object-oriented applications. Orbix provides a full implementation of the Common Object Request Broker Architecture (CORBA) from the Object Management Group (OMG). It is compliant with version 2.4 of the OMG'S CORBA specification. This guide explains how to configure and manage the components of an Orbix environment.

Audience

This guide is intended to be used by system administrators, in conjunction with the *Administrator's Guide*. It assumes that the reader is familiar with Orbix administration.

Organization of this guide

This guide is divided as follows:

- [Chapter 1](#) provides a brief overview of Orbix configuration, how it is organized, and the syntax for specifying variable entries.
- [Chapter 2](#) describes the root namespace of an Orbix configuration and what variables belong in it.
- [Chapter 4](#) describes the configuration namespaces and variables that control the core functionality of Orbix.
- [Chapter 6](#) describes the configuration variables that define a configuration domain
- [Chapter 3](#) describes the configuration namespaces and variables used to configure COMet.
- [Chapter 7](#) describes the configuration namespaces and variables used to configure the Plug-ins to the Adaptive Runtime Technology core. These plug-ins include the CORBA services.

- [Chapter 8](#) describes the configuration variables in the `polcies` namespace.
- [Chapter 9](#) describes the configuration namespaces and variables used to configure IONA's JMS implementation and the JMS-Notification bridge.
- [Chapter 10](#) describes the configuration namespaces and variables used to configure Orbix security features.
- [Chapter 11](#) describes the configuration variables used to configure the XA Resource Manager plug-in.

Related documentation

If you are new to Orbix, it is recommended that you read the *Orbix Administrator's Guide*. This guide provides an overview of the Orbix environment and how to manage an Orbix installation.

The latest updates to the Orbix documentation can be found at <http://www.iona.com/docs/>.

Additional resources

The [IONA knowledge base](http://www.iona.com/support/knowledge_base/index.xml) (http://www.iona.com/support/knowledge_base/index.xml) contains helpful articles, written by IONA experts, about the Orbix and other products.

The [IONA update center](http://www.iona.com/support/updates/index.xml) (<http://www.iona.com/support/updates/index.xml>) contains the latest releases and patches for IONA products.

If you need help with this or any other IONA products, contact IONA at support@iona.com. Comments on IONA documentation can be sent to docs-support@iona.com.

Typographical conventions

This guide uses the following typographical conventions:

`Constant width` Constant width (courier font) in normal text represents portions of code and literal names of items such as classes, functions, variables, and data structures. For example, text might refer to the `CORBA::Object` class.

Constant width paragraphs represent code examples or information a system displays on the screen. For example:

```
#include <stdio.h>
```


Italic

Italic words in normal text represent *emphasis* and *new terms*.

Italic words or characters in code and commands represent variable values you must supply, such as arguments to commands or path names for your particular system. For example:

```
% cd /users/your_name
```

Note: Some command examples may use angle brackets to represent variable values you must supply. This is an older convention that is replaced with *italic* words or characters.

Keying conventions

This guide may use the following keying conventions:

No prompt	When a command's format is the same for multiple platforms, a prompt is not used.
%	A percent sign represents the UNIX command shell prompt for a command that does not require root privileges.
#	A number sign represents the UNIX command shell prompt for a command that requires root privileges.
>	The notation > represents the DOS, Windows NT, Windows 95, or Windows 98 command prompt.
...	Horizontal or vertical ellipses in format and syntax descriptions indicate that material has been eliminated to simplify a discussion.
.	
.	
.	
[]	Brackets enclose optional items in format and syntax descriptions.
{}	Braces enclose a list from which you must choose an item in format and syntax descriptions.
	A vertical bar separates items in a list of choices enclosed in {} (braces) in format and syntax descriptions.

PREFACE

Introduction

An Orbix configuration domain is a collection of configuration information in an Orbix environment. This information consists of configuration variables and their values. Configuration domains are implemented in an Orbix configuration repository or in a configuration file.

In this chapter

This chapter includes the following sections:

Orbix Configuration Concepts	page 2
Configuration Data types	page 4

Orbix Configuration Concepts

Overview

The main concepts and components in an Orbix configuration domain are as follows:

- “Configuration scopes”
 - “ORB name mapping”
 - “Configuration namespaces”
 - “Configuration variables”
-

Configuration scopes

An Orbix configuration is divided into configuration scopes. Applications can have their own configuration scopes, and specific parts of applications (specific ORBs) can have ORB-specific scopes.

Scopes are typically organized into a hierarchy of scopes, whose fully-qualified names map directly to ORB names. By organizing configuration variables into various scopes, you can provide different settings for individual ORBs, or common settings for groups of ORBs.

Configuration scopes apply to a subset of ORBs or to a specific ORB in an environment. Orbix services, such as the locator service, have their own configuration scopes. Orbix service scopes are automatically created when you configure those services into a new domain.

ORB name mapping

An initializing ORB maps to a configuration scope through its ORB name. For example, if an initializing ORB is supplied with a command-line `-ORBname` argument of `company.operations`, it uses all variable settings in that scope, and the parent `company` and root scopes. Settings at narrower scopes such as `company.operations.finance`, and settings in unrelated scopes such as `company.production`, are unknown to this ORB and so have no effect on its behavior.

If an initializing ORB does not find a scope that matches its name, it continues its search up the scope tree. For example, given the hierarchy shown earlier, ORB name `company.operations.finance.payroll` will fail to find a scope that matches. An ORB with that name next tries the parent scope `company.operations.finance`. In this case, ORB and scope names match and the ORB uses that scope. If no matching scope is found, the ORB takes its configuration from the root scope.

Configuration namespaces

Most configuration variables are organized within namespaces, which serve to group related variables. Namespaces can be nested, and are delimited by colons (:). For example, the initial reference for the locator daemon plug-in is specified as follows:

```
initial_references:IT_Locator:reference
```

Configuration variables

The actual configuration data is stored in variables that are set within each namespace. In some instances variables in different namespaces share the same variable names.

Variables can also be reset several times within successive layers of a configuration scope. Configuration variables set in narrower configuration scopes override variable settings in wider scopes. For example, the `company.operations.orb_plugins` variable overrides `company.orb_plugins`. Thus, the plug-ins specified at the `company` scope apply to all ORBs in that scope, except those ORBs that belong specifically to the `company.operations` scope and its child scopes, `hr` and `finance`.

Configuration Data types

Overview

Each configuration variable has an associated data type that determines the variable's value. When creating configuration variables, you must specify the variable type.

Data types can be categorized as follows:

- [Primitive types](#)
- [Constructed types](#)

Primitive types

Orbix supports the following primitive types:

- `boolean`
- `double`
- `long`

These correspond to IDL types of the same name. See the *CORBA Programmer's Guide* for more information.

Constructed types

Orbix supports two constructed types: `string` and `ConfigList` (a sequence of strings).

- A `string` is an IDL string whose character set is limited to the character set supported by the underlying configuration domain type. For example, a configuration domain based on ASCII configuration files could only support ASCII characters, while a configuration domain based on a remote configuration repository might be able to perform character set conversion.

Variables of `string` also support string composition. A composed string variable is a combination of literal values and references to other string variables. When the value is retrieved, the configuration system replaces the variable references with their values, forming a single complete string.

- The `ConfigList` type is simply a sequence of `string` types. For example:

```
orb_plugins = ["local_log_stream", "iiop_profile",  
              "giop","iiop"];
```


Root Namespace

The root namespace includes the following variables:

orb_plugins	page 7
secure_directories	page 8

orb_plugins

The `orb_plugins` variable specifies the plug-ins that the ORB should load during application initialization. A plug-in is a class or code library that can be loaded into an Orbix application at link-time or runtime. These plug-ins provide the user the ability to load network transports, error logging streams, CORBA services, and other features “on the fly.” For more information see [“CORBA Plug-ins” on page 43](#).

The following example variable specifies Orbix error logging, and the transport protocols to use:

```
orb_plugins=["local_log_stream", "iiop_profile", "giop",  
            "iiop"];
```

secure_directories

The `secure_directories` variable specifies a comma-separated list of secure directories in which the node daemon can launch processes. When the node daemon attempts to launch a registered process, it checks its pathname against the `secure_directories` list. If a match is found, the process is activated; otherwise, the node daemon returns a `StartProcessFailed` exception to the client.

For example, the following configuration file entry specifies two secure directories:

```
secure_directories=["c:\Acme\bin,c:\my_app"];
```

COMet

The `COMet` namespaces contain configuration variables that are specific to COMet, and their associated default values.

In this chapter

This chapter discusses the following configuration namespaces:

COMet:Config	page 10
COMet:Mapping	page 12
COMet:Debug	page 13
COMet:TypeMan	page 14
COMet:Services	page 18

COMet:Config

The variables in this namespace control the runtime behavior of the COMet bridge. It contains the following variables:

- [COMET_SHUTDOWN_POLICY](#)
- [SINGLE_THREADED_CALLBACK](#)
- [USE_INTERFACE_IN_IOR](#)

COMET_SHUTDOWN_POLICY

`COMET_SHUTDOWN_POLICY` specifies the shutdown policy for COMet. Set this variable to one of the following values:

<code>implicit</code>	(Default) Specifies that COMet is to be shut down the first time <code>DllCanUnloadNow</code> is about to return <code>yes</code> .
<code>explicit</code>	Specifies that an application must call <code>ORB::ShutDown()</code> to force COMet to shut down.
<code>Disabled</code>	Specifies that COMet does not shut down the ORB when it thinks it is about to unload. That is, the DLL is not unloaded when <code>DllCanUnloadNow</code> is called by the COM runtime. Visual Basic and Internet Explorer do this to cache the DLLs. A problem arises, however, if the DLL is reused, because Orbix has already been shut down.
<code>atExit</code>	Specifies that the COMet bridge is shut down only at process-exit time. This is the recommended setting when running in the Visual Basic development environment.

SINGLE_THREADED_CALLBACK

`SINGLE_THREADED_CALLBACK` is a boolean variable which lets you implement your own event loop for processing callbacks, instead of having COMet dispatch them as they arrive. Defaults to `false`.

USE_INTERFACE_IN_IOR

USE_INTERFACE_IN_IOR is a boolean that causes COMet to use the type id embedded in the IOR as the interface name when performing a narrow thus eliminating the need for any remote calls and thereby reducing IFR network traffic.

COMet:Mapping

The variables in this namespace control how COMet maps from OMG IDL datatypes to COM IDL datatypes. It contains the following variables:

- [SAFEARRAYS_CONTAIN_VARIANTS](#)
- [KEYWORDS](#)

SAFEARRAYS_CONTAIN_VARIANTS

`SAFEARRAYS_CONTAIN_VARIANTS` is a boolean variable which, when set to `true`, enables COMet to determine, when constructing an `out` parameter, whether the parameter type has been declared (using the `dim` statement) as the real type from the type library, or simply as `SAFEARRAY`.

This variable addresses how Visual Basic deals with `SafeArrays` as `out` parameters. Visual Basic does not correctly check the `v_VT` type of the `SafeArray` contents and automatically assumes they are of the `VARIANT` type.

For example, COMet can use this variable's setting to determine whether a sequence of `long` types maps to a `SAFEARRAY` of `long` types, or to a `SAFEARRAY` of `VARIANTS`, where each `VARIANT` contains a `long`.

KEYWORDS

`KEYWORDS` specifies a list of strings to be prefixed with `IT_`, in order to avoid name clashes when using `ts2idl` to generate COM IDL from existing OMG IDL type information in the type store.

COMet:Debug

The variables in this namespace control how COMet logs debug information. It contains the following variables:

- [MessageLevel](#)

MessageLevel

`MessageLevel` specifies how much logging information to make available and the log file's location. This variable's value consists of two comma-delimited fields:

- A value between 0 and 255, inclusive, that specifies the level of verbosity, where 0 specifies to log no messages, and 255 specifies to log all messages.
- The log file's pathname.

For example, the following setting specifies that all messages are logged in `comet.log`:

```
COMet:Debug:MessageLevel="255, c:\temp\comet.log"
```

COMet:TypeMan

The variables in this namespace control the behavior of COMet's type store manager. It contains the following variables:

- [TYPEMAN_CACHE_FILE](#)
- [TYPEMAN_DISK_CACHE_SIZE](#)
- [TYPEMAN_IFR_IOR_FILENAME](#)
- [TYPEMAN_IFR_NS_NAME](#)
- [TYPEMAN_LOG_FILE](#)
- [TYPEMAN_LOGGING](#)
- [TYPEMAN_MEM_CACHE_SIZE](#)
- [TYPEMAN_READONLY](#)

TYPEMAN_CACHE_FILE

`TYPEMAN_CACHE_FILE` specifies the name and location of the cache file that COMet uses to access type information efficiently. This variable is automatically set by the configuration script.

The following example shows the default setting, where *install-dir* represents the Orbix installation directory, and *domain-name* represents the domain name:

```
COMet:TypeMan:TYPEMAN_CACHE_FILE="install-dir\var\domain-name\database\comet"
```

TYPEMAN_DISK_CACHE_SIZE

`TYPEMAN_DISK_CACHE_SIZE` specifies the maximum number of entries allowed in the disk cache, where each entry corresponds to a user-defined type—for example, an IDL union or interface definition. When cache entries exceed this variable's setting, the cache can be flushed.

This variable's setting depends on the nature of applications using the bridge. In general, disk cache size should be about eight to ten times greater than the memory cache (see [TYPEMAN_MEM_CACHE_SIZE](#)).

Given a typical mix of user-defined types, 1000 cache entries use up about 2 megabytes of disk space. Thus, the following setting:

```
COMet:TypeMan:TYPEMAN_DISK_CACHE_SIZE="2000"
```

allows approximately 4 megabytes maximum disk cache file size. When the cache is primed with type libraries for DCOM servers, the size can be considerably larger. The size depends on the size of the type libraries, which can vary considerably. Typically, a primed type library is more than three times the size of the original type library, because the information is stored in a format that optimizes speed.

TYPEMAN_IFR_IOR_FILENAME

When the dynamic marshalling engine in COMet encounters a type for which it cannot find corresponding type information in the type store, it must then retrieve the type information from the interface repository. The order in which COMet attempts to connect to the interface repository is as follows:

1. If a name is specified in `COMet:TypeMan:TYPEMAN_IFR_NS_NAME`, COMet looks up that name in the Naming Service to connect to the Interface Repository.
2. If a name is not specified in `COMet:TypeMan:TYPEMAN_IFR_NS_NAME`, COMet checks to see if an IOR is specified in `initial_references:InterfaceRepository:reference`. If so, it uses the interface repository associated with that IOR.
3. If an IOR is not specified in `initial_references:InterfaceRepository:reference`, COMet checks to see if a filename is specified in `TYPEMAN_IFR_IOR_FILENAME`.

Consequently, you must set the `TYPEMAN_IFR_IOR_FILENAME` variable if you do not set `COMet:TypeMan:TYPEMAN_IFR_NS_NAME` OR `initial_references:InterfaceRepository:reference`. In this case, the value required is the full pathname to the file that contains the IOR for the interface repository you want to use.

TYPEMAN_IFR_NS_NAME

`TYPEMAN_IFR_NS_NAME` identifies the interface repository's name within the naming service. You should register an IOR for the interface repository in the naming service under a compound name. This variable should contain that compound name. As explained in `TYPEMAN_IFR_IOR_FILENAME`, this is the first configuration variable that COMet always checks if it needs to contact the interface repository for type information that it cannot find in the type store.

TYPEMAN_LOG_FILE

`TYPEMAN_LOG_FILE` specifies the path to the output file for `typeman` logging information, used if `TYPEMAN_LOGGING` is set to `file`.

TYPEMAN_LOGGING

`TYPEMAN_LOGGING` specifies how to output logging information for the COMet type store manager. Set this variable to one of the following values:

None	Default.
stdout	Use only with <code>typeman.exe</code> .
DBMon	Sends output to <code>DBMon.exe</code> .
file	Sends output to the file specified by <code>COMet:Typeman:TYPEMAN_LOG_FILE</code> .

TYPEDMAN_MEM_CACHE_SIZE

`TYPEDMAN_MEM_CACHE_SIZE` specifies the maximum number of entries allowed in the memory cache, where each entry corresponds to a user-defined type—for example, an IDL union or interface definition. When cache entries exceed this variable's setting, the cache can be flushed.

To avoid unnecessary disk swapping, set this variable to at least 100.

TYPEDMAN_READONLY

`TYPEDMAN_READONLY` is a boolean variable which specifies whether read-only mode is used for the type store.

COMet:Services

The variables in this namespace control which service instances the COMet bridge accesses. It contains the following variables:

- [NameService](#)

NameService

`NameService` instructs COMet to use a different naming service than the one specified in `initial_references:NameService`. The value specified is the full pathname to the file that contains the IOR for the desired naming service.

Core Namespaces

The Orbix core services are configured using a number of variables in different namespaces.

In this chapter

This chapter discusses the following configuration variable namespaces:

initial_references	page 20
binding	page 23
domain_plugins	page 27
event_log	page 28
orb_management	page 29
poa:fqpn	page 30
thread_pool	page 32
url_resolvers	page 35

initial_references

The `initial_references` namespace contains a child namespace for each initial reference available to Orbix. Child namespaces have the same name as the referenced service. For example:

```
initial_references:InterfaceRepository
initial_references:ConfigRepository
initial_references:DynAnyFactory
```

Each child namespace contains a variable called `plugin` or `reference`.

- If the variable is `reference`, its value is an IOR. For example:

```
initial_references:IT_Locator:reference =
    "IOR:010000002....";
```

- If the variable is `plugin`, its value is the plugin that provides the reference. For example:

```
initial_references:RootPOA:plugin = "poa";
```

All domain services, such as the locator daemon, interface repository, and naming service, must have their initial object references set in the configuration's root configuration scope. For example, in a file-based configuration, the following entry sets the locator daemon's initial reference:

```
initial_references:IT_Locator:reference = "IOR:200921....";
```

IT_CodeSet_Registry:plugin

`IT_CodeSet_Registry:plugin` specifies the codeset conversion library to load. The default CodeSet Plugin contains full codeset conversion functionality. However, this conversion library is over 8MB in size. Therefore, users who do not require full codeset conversion functionality may choose to load the smaller basic codeset conversion library.

The name of the full codeset conversion library is `codeset`. The name of the smaller basic codeset conversion library is `basic_codeset`.

Note: The Java ORB will load the full codeset conversion library regardless of what setting you choose.

For more information on these plugins, refer to the *Internationalization Guide*.

IT_CSI:plugin

`IT_CSI:plugin` specifies the plugin for Common Secure Interoperability (CSI). The default value is: `initial_references:IT_CSI:plugin = "csi";` For more details, see the *Security Guide*.

IT_JMSMessageBroker:reference

`IT_JMSMessageBroker:reference` specifies the object reference of the JMS broker.

IT_JMSServerContext:reference

`IT_JMSServerContext:reference` supports JNDI lookup of JMS destinations and connection factories.

OTSMangement:plugin

`OTSMangement:plugin` specifies the plugin that provides the management functionality for the plugin that supports the `TransactionService` IDL interface. If no plugin is specified, the OTS server runs unmanaged.

TransactionFactory:plugin

`TransactionFactory:plugin` specifies the plugin that supports the `TransactionFactory` IDL interface. This plugin is loaded on demand in response to invocations of `resolve_initial_references("TransactionFactory")`.

TransactionFactory:reference

`TransactionFactory:reference` specifies the object references (as a URL) of a server that supports the `TransactionFactory` IDL interface. This variable is used when a standalone transaction manager service is used. This variable takes precedence over `initial_references:TransactionFactory:plugin`.

TransactionCurrent:plugin

`TransactionCurrent:plugin` specifies the plugin that supports the `TransactionCurrent` IDL interface. For example:

```
initial_references:TransactionCurrent:plugin="ots";
```

TransactionManager:plugin

`TransactionManager:plugin` specifies the plugin that supports the `TransactionManager` IDL interface. For example:

```
initial_references:TransactionManager:plugin="jta_manager";
```

UserTransaction:plugin

`UserTransaction:plugin` specifies the plugin that supports the `UserTransaction` IDL interface. For example:

```
initial_references:UserTransaction:plugin="jta_user";
```

binding

The `binding` namespace contains variables that specify interceptor settings. Orbix uses interceptors internally to process requests. In CORBA a *binding* is a set of interceptors used to process requests. Orbix creates both client-side and server-side bindings, at request-level and message-level, for CORBA applications. Client-side bindings and request-level server-side bindings are created at POA granularity.

On both the client and server sides, interceptors listed in the binding list can decide that they are not needed. This is based on the effective policies, or the IOR profile used, or both. If interceptors are not needed, the binding is created with the other listed interceptors.

The `binding` namespace includes the following variables:

- `client_binding_list`
- `server_binding_list`
- `servlet_binding_list`

client_binding_list

Orbix provides client request-level interceptors for OTS, GIOP, and POA collocation (where server and client are collocated in the same process). Orbix provides message-level interceptors used in client-side bindings for IIOP, SHMIOP and GIOP.

`client_binding_list` specifies a list of potential client-side bindings. Each item is a string that describes one potential interceptor binding. For example:

```
[ "OTS+POA_Colloc", "POA_Colloc", "OTS+GIOP+SHMIOP", "GIOP+SHMIOP",  
  "OTS+GIOP+IIOP", "GIOP+IIOP" ];
```

Interceptor names are separated by a plus (+) character. Interceptors to the right are closer to the wire than those on the left. The syntax is as follows:

- Request-level interceptors, such as `GIOP`, must precede message-level interceptors, such as `IIOP`.

- `GIOP` or `POA_coloc` must be included as the last request-level interceptor.
- Message-level interceptors must follow the `GIOP` interceptor, which requires at least one message-level interceptor.
- The last message-level interceptor must be a message-level transport interceptor, such as `IIOP` or `SHMIOP`.

When a client-side binding is needed, the potential binding strings in the list are tried in order, until one successfully establishes a binding. Any binding string specifying an interceptor that is not loaded, or not initialized through the `orb_plugins` variable, is rejected.

For example, if the `ots` plugin is not configured, bindings that contain the `OTS` request-level interceptor are rejected, leaving `["POA_Coloc", "GIOP+IIOP", "GIOP+SHMIOP"]`. This specifies that POA collocations should be tried first; if that fails, (the server and client are not collocated), the `GIOP` request-level interceptor and the `IIOP` message-level interceptor should be used. If the `ots` plugin is configured, bindings that contain the `OTS` request interceptor are preferred to those without it.

server_binding_list

`server_binding_list` specifies interceptors included in request-level binding on the server side. The POA request-level interceptor is implicitly included in the binding.

The syntax is similar to `client_binding_list`. However, the left-most interceptors are closer to the wire, and no message-level interceptors can be included (for example, `IIOP`). An empty string (`"`) is a valid server-side binding string. This specifies that no request-level interceptors are needed. A binding string is rejected if any named interceptor is not loaded and initialized.

The default `server_binding_list` is `["OTS", ""]`. If the `ots` plugin is not configured, the first potential binding is rejected, and the second potential binding (`"`) is used, with no explicit interceptors added.

servlet_binding_list

`servlet_binding_list` specifies a list of potential servlet bindings. For example:

```
binding:servlet_binding_list=["it_servlet_context +
    it_naming_context + it_exception_mapping + it_http_sessions +
    it_web_security + it_servlet_filters +
    it_web_app_activator"];
```

buffer

The `buffer` namespace contains information used by the ORB's buffer manager.

heap_storage_size

`heap_storage_size` defines the size of memory blocks allocated by the ORB's buffer manager on the heap. This is available in the Java and C++ ORB.

domain_plugins

The `domain_plugins` namespace contains information about the plugins required to access the configuration domain. For example, a domain of `itconfig://IOR000123...` uses the `cfr_handler` plugin to contact the configuration repository:

```
domain_plugins:itconfig = "cfr_handler";
```

event_log

This namespace control the logging of Orbix subsystems, such as POAs and services. It contains the following variables:

- `filters`

filters

`filters` sets the level of logging for specified subsystems, such as POAs, or the naming service. This variable specifies a list of filters, where each filter sets logging for a specified subsystem, with the following format:

```
subsystem=severity-level[+severity-level]...
```

For example, the following filter instructs the Orbix to report only errors and fatal errors for the naming service:

```
IT_NAMING=ERR+FATAL
```

The subsystem field indicates the name of the Orbix subsystem that reports the messages. The severity field indicates the severity levels that are logged by that subsystem.

The following entry in a configuration file explicitly sets message severity levels for the POA and ORB core, and all other subsystems:

```
event_log:filters = ["IT_POA=INFO_HI+WARN+ERROR+FATAL",  
"IT_CORE=*", "*=WARN+ERR+FATAL"];
```

For more information about using this variable, see the *Orbix Administrator's Guide*.

orb_management

The variable in this namespace configures ORB management.

- `retrieve_existing_orb`

retrieve_existing_orb

`retrieve_existing_orb` only controls the behavior of Java-based CORBA applications. It determines if calls to `ORB.init()` can return an existing ORB instance. Under the standard IDL-to-Java mapping, each call to `ORB.init()` returns a new ORB instance for use in applications. This conflicts with the C++ mapping of `ORB_init()`, where an existing ORB can be returned, when identified using the `-ORBid` argument.

If the `retrieve_existing_orb` variable is set to `true` in an ORB-specific configuration scope, Orbix allows an existing ORB to be returned by `ORB.init()`. This prevents applications from inadvertently creating several ORB instances. If this variable is set to `false`, and an attempt is made to retrieve an existing ORB, a `CORBA:NO_PERMISSION` exception is raised. Defaults to `false`.

poa:fqpn

Orbis has two configuration variables that allow POAs to use direct persistence and well-known addressing if the policies have not been set programatically. Both variables specify the policy for individual POAs by specifying the fully qualified POA name for each POA. They take the form:

```
poa:fqpn:variable
```

For example to set the well-known address for a POA whose fully qualified POA name is `darleen` you would set the variable

```
poa:darleen:well_known_address.
```

The following variables are in this namespace:

- `direct_persistent`
- `well_known_address`

direct_persistent

`direct_persistent` specifies if a POA runs using direct persistence. If this is set to `true` the POA generates IORs using the well-known address that is specified in the `well_known_address` variable. Defaults to `false`. For an example of how this works, see [well_known_address](#).

well_known_address

`well_known_address` specifies the address used to generate IORs for the associated POA when that POA's `direct_persistent` variable is set to `true`.

For example, by default, the `simple_persistent` demo creates an indirect persistent POA called `simple_persistent`. If you want to run this server using direct persistence, and well known addressing, add the following to your configuration:

```
simple_orb {  
    poa:simple_persistent:direct_persistent = "true";  
    poa:simple_persistent:well_known_address = "simple_server";  
    simple_server:iiop:port = "5555";  
};
```


All object references created by the `simple_persistent` POA will now be direct persistent containing the well known IIOP address of port 5555.

Obviously, if your POA name was different the configuration variables would need to be modified. The scheme used is the following:

```
poa:<FQPN>:direct_persistent=<BOOL>;  
poa:<FQPN>:well_known_address=<address_prefix>;  
<address_prefix>:iiop:port=<LONG>;
```

<FQPN> is the fully qualified poa name. Obviously this introduces the restriction that your poa name can only contain printable characters, and may not contain white space.

<address_prefix> is the string that gets passed to the well-known addressing POA policy. Specify the actual port used using the variable <address_prefix>:iiop:port. You can also use `iiop_tls` instead of `iiop`.

Note: This functionality is currently only implemented in the C++ ORB. If you are using the Java ORB, you must set the direct persistence and well known addressing policies programmatically.

thread_pool

The variables in the `thread_pool` namespace specify policies that configure multi-threading. This namespace includes the following variables:

- `high_water_mark`
 - `initial_threads`
 - `low_water_mark`
 - `max`
 - `max_queue_size`
 - `stack_size`
-

high_water_mark

`high_water_mark` specifies the maximum number of threads allowed in the thread pool. Defaults to `-1`, which means that there is no limit on the maximum number of threads.

For C++ processes, you must ensure that the `high_water_mark` thread limit does not exceed any OS-specific thread limit (for example, `nkthreads` or `max_thread_proc`). Otherwise, thread creation failure would put your process into an undefined state.

In general, for Java processes (JDK 1.3.x), you should prevent the ORB from reaching the `high_water_mark` thread limit. This is because the Java ORB uses a thread-per-connection approach due to limitations in the JDK 1.3.x socket implementation.

initial_threads

`initial_threads` specifies the number of initial threads in the thread pool. Defaults to the `low_water_mark` thread limit (or 5, if the `low_water_mark` is not set).

low_water_mark

`low_water_mark` specifies the minimum number of threads in the thread pool. If this variable is set, the ORB will terminate unused threads until only this number exists. The ORB can then create more threads, if needed, to handle the items in its work queue.

Defaults to `-1`, which means do not terminate unused threads.

Note: The Java ORB requires at least 4 worker threads to correctly dispatch requests. Attempting to restrict the thread pool to less than four threads will cause Java clients to hang.

max

`max` sets the maximum number of threads that are available for JMS message processing.

max_queue_size

`max_queue_size` specifies the maximum number of request items that can be queued on the ORB's internal work queue. If this limit is exceeded, Orbix considers the server to be overloaded, and gracefully closes down connections to reduce the load. The ORB will reject subsequent requests until there is free space in the work queue.

Defaults to `-1`, which means that there is no upper limit on the size of the request queue. In this case, the maximum work queue size is limited by how much memory is available to the process.

There is no direct relationship between `max_queue_size` and `high_water_mark`. A particular value for `high_water_mark` does not require a corresponding value for `max_queue_size`. For example, even if the queue size is unbounded, each work item should be serviced eventually by the ORB's available threads. However, this will not occur if the threads are hung up indefinitely and unable to execute a new request from the work queue.

You can also install your own `AutomaticWorkQueue` for a POA to use in your server, where you define the limits for your queue programmatically. In a `ManualWorkQueue`, you must code the threads that pull items from the queue. The only programmatic variable you control for a `ManualWorkQueue` is maximum queue size. See the *Orbix Programmer's Guide* for more details.

stack_size

`stack_size` sets the ORB's internal threads stack size.

This is only available in the C++ ORB; the Java API does not allow manipulation of the thread stack size.

url_resolvers

This namespace contains variables that determine how to resolve interoperable naming URLs. For example, the following variable specifies that the `naming_resolver` plugin should be used for the `corbaname` resolver:

```
url_resolvers:corbaname:plugin = "naming_resolver";
```

The following variable specifies the library for the `naming_resolver` plugin:

```
plugins:naming_resolver:shlib_name = "it_naming";
```

The following variable specifies the library for the `naming_resolver` plugin:

```
plugins:naming_resolver:ClassName =  
    "com.ionacorba.naming_resolver.CORBANamePlugIn";
```

The following interoperable naming URL causes the `naming_resolver` plugin to be loaded:

```
corbaname::555xyz.com/dev/NContext1#a/b/c
```

The `naming_resolver` plugin is then used to resolve the URL.

ClassLoader

This chapter describes the configuration variables used to control Java classloading.

In this chapter

This chapter contains the following variables:

cache_url	page 38
jarcache_low_watermark	page 38
jarcache_high_watermark	page 38
use_single_classloader	page 39
force_explode_wars_to_disk	page 39
use_single_classloader_for_webinf	page 39
jar_dependency_list	page 40
cache_scrub_time	page 40

classloader

A Java classloader is a part of the Java virtual machine (JVM) that finds and loads Java class files into memory at runtime. This chapter describes the configuration variables that control Java classloading.

cache_url

`cache_url` specifies the directory on the local file system where the classloading cache is stored. The default value is:

```
CLASSLOADING_CACHE_URL :  
"file:///D:\VAR_DIR\domains\<<domain_name>\cache";
```

jarcache_low_watermark

JAR libraries are cached on disk or in memory. These watermark settings are used to decide whether a JAR is cached on disk or in memory:

- If a JAR is smaller than `jarcache_low_watermark`, it is cached in memory. If a JAR is bigger than `jarcache_high_watermark`, it is cached on disk.
- If a JAR is between the low and high watermark, it is cached in memory if there is adequate memory still available to the JVM.
- Otherwise it is cached on disk.

The default value for `jarcache_low_watermark` is 131072 (128K).

jarcache_high_watermark

JAR libraries are cached on disk or in memory. These watermark settings are used to decide whether a JAR is cached on disk or in memory:

- If a JAR is smaller than `jarcache_low_watermark`, it is cached in memory. If a JAR is bigger than `jarcache_high_watermark`, it is cached on disk.

- If a JAR is between the low and high watermark, it is cached in memory if there is adequate memory still available to the JVM.
- Otherwise it is cached on disk.

The default value for `jarcache_high_watermark` is `262144` (256K).

use_single_classloader

`use_single_classloader` specifies either:

- a single classloader per application. (`true`)
- a single classloader per module. (`false`)

The default value is `true`.

force_explode_wars_to_disk

This setting indicates whether or not WAR files are always extracted to disk. This is required by certain web applications that need direct file-based I/O access to their own resources. Setting this value to `false` gives the application server the possibility to extract the archive into memory which may improve performance and save disk space. In this case, the decision to extract to memory or disk is dependent on the `jarcache_low_watermark` and the `jarcache_high_watermark` settings.

use_single_classloader_for_webinf

`use_single_classloader_for_webinf` specifies either:

- a single classloader for the contents of the `web-inf` library. (`true`)
- a single classloader per `.jar` file. (`false`)

Although a single classloader for all of the JARs in the `web-inf` lib is compliant with the J2EE specification, a classloader per JAR may be more memory efficient. This configuration item is only useful when using a classloader per module. The default value is `true`.

jar_dependency_list

When using a classloader per module, it is necessary to specify any JAR dependencies that are not explicitly mentioned in the manifest `CLASSPATH` of a JAR. For example, if your application uses a `util.jar` that in turn uses an `extlib.jar`, this `util.jar` must either mention the `extlib.jar` in its manifest `CLASSPATH` (preferred) or enter it here in the `jar_dependency_list`.

For example:

```
ipas:classloader:jar_dependency_list=["jdom.jar=xerces.jar",  
  "MyApp.jar=lib1.jar,lib2.jar"];
```

The default here is: ["jdom.jar=xerces.jar"]

cache_scrub_time

`cache_scrub_time` specifies the classloader scrubbing time. Those archives not used within this time are removed from the cache. The default is 20160 minutes.

Note: These configuration variables apply to all server instances.

Configuration Namespace

The `configuration` namespace contains variables which identify a configuration domain.

In this chapter

This chapter discusses the following configuration variables:

configuration	page 42
domain_dir	page 42

configuration

The configuration namespace includes the following configuration domain-specific variables:

- `domain_name`
 - `domain_dir`
-

domain_name

`domain_name` is the text name used to identify the current domain.

You can set an application's domain with the `-ORBdomain_name` parameter. For C++ applications, you can also set the `IT_DOMAIN_NAME` environment variable. For more information, see the *Orbix Administrator's Guide*.

domain_dir

`domain_dir` specifies the location of your configuration domain files.

You can set this location using the `-ORBconfig_domains_dir` parameter; For C++ applications, you can also set the `IT_CONFIG_DOMAINS_DIR` environment variable. For more information, see the *Orbix Administrator's Guide*.

CORBA Plug-ins

Orbix is built on IONA's Adaptive Runtime architecture (ART), which enables users to configure services as plugins to the core product.

Overview

A plugin is a class or code library that can be loaded into an Orbix application at link-time or runtime. The `plugins` namespace contains child namespaces for plugins, such as `naming` and `iiop`. Each child namespace has information specific to each plugin. Child namespaces usually have a Java `ClassName` or C++ `shlib_name` variable, indicating the class or library in which the plugin resides. The following examples show how the configuration specifies the library or class name for the `iiop` plugin:

C++

```
plugins:iiop:shlib_name = "it_iiop";
```

Java

```
plugins:iiop:ClassName="com.iona.corba.iiop.IIOPPlugIn";
```

Plugins also have their own specific configuration variables. For example, the following variable sets the default timeout of a transaction in seconds:

```
plugins:ots:default_transaction_timeout
```

In this chapter

The following plugins are discussed in this chapter:

plugins:atli2_ip	page 46
plugins:atli2_shm	page 47
plugins:basic_log	page 49
plugins:codeset	page 50
plugins:config_rep	page 54
plugins:connection_filter	page 55
plugins:egmiop	page 56
plugins:event	page 58
plugins:event_log	page 62
plugins:giop	page 63
plugins:giop_snoop	page 64
plugins:http(s)	page 67
plugins:i18n	page 71
plugins:iiop	page 73
plugins:ifr	page 78
plugins:it_http_sessions	page 80
plugins:it_mgmt	page 81
plugins:it_mbean_monitoring	page 82
plugins:it_pluggable_http_sessions	page 83
plugins:it_response_time_collector	page 85
plugins:it_security_service	page 87
plugins:file_security_domain	page 88
plugins:jta	page 89

plugins:local_log_stream	page 91
plugins:locator	page 93
plugins:naming	page 96
plugins:node_daemon	page 99
plugins:notify	page 101
plugins:notify:database	page 105
plugins:notify_log	page 109
plugins:orb	page 110
plugins:ots	page 111
plugins:ots_lite	page 116
plugins:ots_encina	page 118
plugins:ots_mgmt	page 125
plugins:poa	page 127
plugins:pss	page 128
plugins:pss_db:envs:env-name	page 129
plugins:pss_db:envs:env-name:dbs:storage-home-type-id	page 141
plugins:shmiop	page 144
plugins:tlog	page 145
plugins:tlog:database	page 148
plugins:ziop	page 152

plugins:atli2_ip

This namespace includes the following:

- [ClassName](#)
- [nio:allocate_heap_byte_buffer](#)

ClassName

`classname` specifies whether the transport layer implementation (ATLI2) uses Java classic I/O (CIO) or new I/O (NIO). The default is CIO.

ATLI2/Java NIO allows more connections to be managed with fewer threads, and also performs better than ATLI2/Java CIO in the presence of many incoming connections.

To enable Java NIO, change the `plugins:atli2_ip:ClassName` configuration variable setting from the following:

```
plugins:atli2_ip:ClassName
=com.ionacorba.atli2.ip.cio.ORBPlugInImpl
```

to the following:

```
plugins:atli2_ip:ClassName
=com.ionacorba.atli2.ip.nio.ORBPlugInImpl
```

For more information on ATLI2/Java NIO, see the *Orbix Administrator's Guide*.

nio:allocate_heap_byte_buffer

`nio:allocate_heap_byte_buffer` specifies whether to use heap buffers or native buffers (the default). To use heap buffers, set `plugins:atli2_ip:nio:allocate_heap_byte_buffer` to `true`.

plugins:atli2_shm

The variables in this namespace control the behavior of the shared memory ATLI2 plugin. This namespace includes the following:

- `max_buffer_wait_time`
- `shared_memory_segment_basename`
- `shared_memory_size`
- `shared_memory_segment`

max_buffer_wait_time

`max_buffer_wait_time` specifies the maximum wait time on a shared memory buffer before raising a no resources exception. The default is 5 seconds.

shared_memory_segment_basename

`shared_memory_segment_basename` defines the prefix used when the shared memory transport creates internal files (for example, in `/var/tmp/SAMD` and `/tmp` on Solaris). The default is `iona`.

shared_memory_size

`shared_memory_size` specifies the size of the shared memory segment created (for example, in the call to `mmap` on Solaris). The default value is $8*1024*1024$.

This size should be larger than the largest data payload passed between a client and server. If the setting is too small, the shared memory transport will run out of memory, and will be unable to marshal the data. If there is danger of this occurring, add `GIOP+IIOP` to your `client_binding_list` setting. This enables the ORB to use the normal network transport if a large payload can not make it through shared memory.

shared_memory_segment

`shared_memory_segment` specifies the name of the already existing shared memory segment to use in place of creating a new segment. There is no default name. Orbix creates a new segment by default.

plugins:basic_log

The variables in this namespace control the behavior of basic log service. These variables include the following:

- `advertise_services`
- `is_managed`
- `shlib_name`

advertise_services

`advertise_services` specifies whether the `basic_log` service should register plain text keys for the object references it publishes in prepare mode. Defaults to `true`.

is_managed

`is_managed` specifies whether or not the basic log service can be managed using the management service. Defaults to `false`, which means the management service does not manage the service.

shlib_name

`shlib_name` identifies the shared library (or DLL in Windows) containing the plugin implementation. The basic log plugin is associated with the base name of the shared library (`it_basic_log_svr` in this case). This library base name is expanded in a platform-dependent manner to obtain the full name of the library file.

```
plugins:basic_log:shlib_name = "it_basic_log_svr";
```

plugins:codeset

The variables in this namespace specify the codesets used by the CORBA portion of Orbix. This is useful when internationalizing your environment.

The following variables are contained in this namespace:

- `plugins:egmiop`
- `interop_allow_null_strings`
- `char:ncs`
- `char:ccs`
- `wchar:ncs`
- `wchar:ccs`

always_use_default

`always_use_default` specifies whether hardcoded default values are used. This means that any `codeset` configuration variables are ignored if they are in the same configuration scope or higher. To enable hardcoded default values, set this variable as follows:

```
plugins:codeset:always_use_default = "true"
```

interop_allow_null_strings

`interop_allow_null_strings` specifies whether to allow `null strings` to be passed. Passing null strings is not CORBA compliant, however, this feature is provided to enable interoperability with third-party software that is not so CORBA compliant. To allow null strings to be passed, set this variable as follows:

```
plugins:codeset:interop_allow_null_strings = "true";
```

This defaults to `false` for CORBA compliance. If this configuration variable is not set, or is set to `false`, and you attempt to pass a `null string`, an exception is thrown. `interop_allow_null_strings` is equivalent to `IT_MARSHAL_NULLS_OK` in Orbix 3.3

Note: Orbix does not support `wstring` null strings with GIOP 1.2 because the CORBA 3.0 specification does not determine the difference between empty strings and null `wstrings`. In this case, the normal exceptions are thrown.

char:ncs

`char:ncs` specifies the native codeset to use for narrow characters. The default setting is determined as follows:

Table 1: *Defaults for the native narrow codeset*

Platform/Locale	Language	Setting
non-MVS, Latin-1 locale	C++	ISO-8859-1
MVS	C++	EBCDIC
ISO-8859-1/Cp-1292/US-ASCII locale	Java	ISO-8859-1
Shift_JS locale	Java	UTF-8
EUC-JP locale	Java	UTF-8
other	Java	UTF-8

char:ccs

`char:ccs` specifies the list of conversion codesets supported for narrow characters. The default setting is determined as follows:

Table 2: *Defaults for the narrow conversion codesets*

Platform/Locale	Language	Setting
non-MVS, Latin-1 locale	C++	
MVS	C++	IOS-8859-1

Table 2: *Defaults for the narrow conversion codesets*

Platform/Locale	Language	Setting
ISO-8859-1/Cp-1292/US-ASCII locale	Java	UTF-8
Shift_JIS locale	Java	Shift_JIS, euc_JP, ISO-8859-1
EUC-JP locale	Java	euc_JP, Shift_JIS, ISO-8859-1
other	Java	file encoding, ISO-8859-1

wchar:ncs

`wchar:ncs` specifies the native codesets supported for wide characters. The default setting is determined as follows:

Table 3: *Defaults for the wide native codesets*

Platform/Locale	Language	Setting
non-MVS, Latin-1 locale	C++	UCS-2, UCS-4
MVS	C++	UCS-2, UCS-4
ISO-8859-1/Cp-1292/US-ASCII locale	Java	UTF-16
Shift_JIS locale	Java	UTF-16
EUC-JP locale	Java	UTF-16
other	Java	UTF-16

wchar:ccs

`wchar:ccs` specifies the list of conversion codesets supported for wide characters. The default setting is determined as follows:

Table 4: *Defaults for the narrow conversion codesets*

Platform/Locale	Language	Setting
non-MVS, Latin-1 locale	C++	UTF-16
MVS	C++	UTF-16
ISO-8859-1/Cp-1292/US-ASCII locale	Java	UCS-2
Shift_JIS locale	Java	UCS-2, Shift_JIS,euc_JP
EUC-JP locale	Java	UCS-2, euc_JP, Shift_JIS
other	Java	file encoding, UCS-2

plugins:config_rep

The `plugins:config_rep` namespace is used to specify high availability settings for the configuration repository (CFR). It includes the following variable:

- `refresh_master_interval`.

refresh_master_interval

`refresh_master_interval` specifies the maximum number of seconds that a slave CFR replica waits for a new master to be declared.

A new master is declared after a failed attempt to delegate an operation to the current master. If no master is found during the specified interval of time, a `TRANSIENT` exception is raised. Defaults to 60.

For example:

```
plugins:config_rep:refresh_master_interval = "40";
```

plugins:connection_filter

The `connection_filter` namespace allow you to select a message-level interceptor plugin.

The plugin closes the connection on locator requests from hosts other than those specified. It will do this until a node daemon has registered - at this point the filter is switched off. The clients need to be able to handle the `CORBA::Exception` as a result of the connection being closed.

This namespace contains the following variables:

- `shlib_name`

shlib_name

This variable is used to specify the plugin library `it_connection_filter`.

For details of how to configure the interceptor see the "Advanced Configuration" chapter in the *Orbix Administrator's Guide*.

plugins:egmiop

The variables in this namespace configure endpoint functionality for the MIOP transport. This namespace contains the following variables:

- `ip:send_buffer_size`
- `ip:receive_buffer_size`
- `pool:java_max_threads`
- `pool:java_min_threads`
- `pool:max_threads`
- `pool:min_threads`
- `udp:packet_size`

ip:send_buffer_size

`ip:send_buffer_size` specifies the `SO_SNDBUF` socket options to control how the IP stack adjusts the size of the output buffer. Defaults to 0, meaning the that buffer size is static.

ip:receive_buffer_size

`ip:receive_buffer_size` specifies the `SO_RCVBUF` socket options to control how the IP stack adjusts the size of the input buffer. Defaults to 0, meaning the buffer size is static.

pool:java_max_threads

`pool:java_max_threads` specifies the maximum number of threads reserved from the `WorkQueue` to support tasks working on behalf of the Java ATLI transport. Defaults to 512.

pool:java_min_threads

`pool:java_min_threads` specifies the minimum number of threads reserved from the `WorkQueue` to support tasks working on behalf of the Java ATLI transport. Defaults to 10.

pool:max_threads

`pool:max_threads` specifies the maximum number of threads reserved from the `WorkQueue` to support tasks working on behalf of the ATLI transport. Defaults to 5.

pool:min_threads

`pool:min_threads` specifies the minimum number of threads reserved from the `WorkQueue` to support tasks working on behalf of the ATLI transport. Defaults to 1.

udp:packet_size

`udp:packet_size` specifies the maximum size for outgoing UDP packets. A larger UDP packet size increases the probability of IP packet fragmentation on the wire hence increasing the possibility of data loss. A smaller UDP packet size increases the overhead per packet and decreases throughput. Defaults to 120 KB.

plugins:event

Overview

The following event service variables are contained in this namespace:

- `advertise_services`
- `direct_persistence`
- `event_pull_interval`
- `max_proxy_consumer_retries`
- `max_proxy_retries`
- `max_proxy_supplier_retries`
- `max_queue_length`
- `operation_timeout_interval`
- `proxy_consumer_retry_delay`
- `proxy_consumer_retry_multiplier`
- `proxy_inactivity_timeout`
- `proxy_retry_delay`
- `proxy_reap_frequency`
- `proxy_retry_multiplier`
- `proxy_supplier_retry_delay`
- `proxy_supplier_retry_multiplier`
- `trace:events`
- `trace:lifecycle`

advertise_services

`advertise_services` specifies whether the event service should register plain text keys for the object references it publishes in prepare mode. Defaults to true.

direct_persistence

`direct_persistence` specifies if the service runs using direct or indirect persistence. The default value is `FALSE`, meaning indirect persistence.

event_pull_interval

`event_pull_interval` specifies the number of milliseconds between successive calls to `pull` on `PullSupplier`. Default value is 1 second.

max_proxy_consumer_retries

`max_proxy_consumer_retries` specifies the maximum number of times to retry before giving up and disconnecting the proxy consumer. If this property is not specified, then the value of `plugins:event:max_proxy_retries` is used.

max_proxy_retries

`max_proxy_retries` specifies the maximum number of times to retry before giving up and disconnecting the proxy. The default value is 3.

max_proxy_supplier_retries

`max_proxy_supplier_retries` specifies the maximum number of times to retry before giving up and disconnecting the proxy supplier. If this property is not specified, then the value of `plugins:event:max_proxy_retries` is used.

max_queue_length

`max_queue_length` specifies the maximum number of events in each event queue. If this limit is reached and another event is received, the oldest event is discarded. The default value is 4096.

operation_timeout_interval

`operation_timeout_interval` specifies the amount of time (in hundreds of nanoseconds) permitted for a blocking request on a client to return before a timeout. The default value is 2 minutes.

proxy_consumer_retry_delay

`proxy_consumer_retry_delay` specifies the initial amount of time in milliseconds that the service waits between successive proxy consumer retries. If this property is not specified, then the value of `plugins:event:proxy_retry_delay` is used.

proxy_consumer_retry_multiplier

`proxy_consumer_retry_multiplier` specifies a double that defines the factor by which the `plugins:event:proxy_consumer_retry_delay` property should be multiplied for each successive proxy consumer retry. If this property is not specified, then the value of `plugins:event:proxy_retry_multiplier` is used.

proxy_inactivity_timeout

`proxy_inactivity_timeout` specifies those proxies that are inactive for the specified number of seconds and disconnects them. The default value is 4 hours, specified in seconds.

proxy_retry_delay

`proxy_retry_delay` specifies the initial amount of time in milliseconds that the service waits between successive retries. The default value is 1 second.

proxy_reap_frequency

`proxy_reap_frequency` specifies the frequency (in seconds) in which inactive proxies are disconnected. The default value is 30 minutes. Setting this property to 0 disables the reaping of proxies.

proxy_retry_multiplier

`proxy_retry_multiplier` specifies a double that defines the factor by which the `retry_delay` property should be multiplied for each successive retry. The default value is 1.

proxy_supplier_retry_delay

`proxy_supplier_retry_delay` specifies the initial amount of time in milliseconds that the service waits between successive proxy supplier retries. If this property is not specified, then the value of `plugins:event:proxy_retry_delay` is used.

proxy_supplier_retry_multiplier

`proxy_supplier_retry_multiplier` specifies a double that defines the factor by which the `plugins:event:proxy_supplier_retry_delay` property should be multiplied for each successive proxy supplier retry. If this property is not specified, then the value of `plugins:event:proxy_retry_multiplier` is used.

trace:events

`trace:events` specifies the output level for event diagnostic messages logged by the service. The default level is 0, which produces no output. A level of 1 or higher produces event processing information and a level of 2 or higher produces event creation and destruction information.

trace:lifecycle

`trace:lifecycle` specifies the output level for lifecycle diagnostic messages logged by the service. The default level is 0, which produces no output. A level of 1 or higher produces lifecycle information (e.g. creation and destruction of Suppliers and Consumers).

plugins:event_log

The variables in this namespace control the behavior of event log service. These variables include the following:

- `advertise_services`
- `is_managed`
- `shlib_name`

advertise_services

`advertise_services` specifies whether the event_log service should register plain text keys for the object references it publishes in prepare mode. Defaults to `true`.

is_managed

`is_managed` specifies whether or not the event log service can be managed using the management service. Defaults to `false`, which means the management service does not manage the service.

shlib_name

`shlib_name` identifies the shared library (or DLL in Windows) containing the plugin implementation. The event log plugin is associated with the base name of the shared library (`it_event_log_svr` in this case). This library base name is expanded in a platform-dependent manner to obtain the full name of the library file.

```
plugins:basic_log:shlib_name = "it_event_log_svr";
```

plugins:giop

This namespace contains the `plugins:giop:message_server_binding_list` configuration variable, which is one of the variables used to configure bidirectional GIOP. This feature allows callbacks to be made using a connection opened by the client, instead of requiring the server to open a new connection for the callback.

message_server_binding_list

`plugins:giop:message_server_binding_list` specifies a list message interceptors that are used for bidirectional GIOP. On the client-side, the `plugins:giop:message_server_binding_list` must be configured to indicate that an existing outgoing message interceptor chain may be re-used for an incoming server binding, similarly by including an entry for `BiDir_GIOP`, for example:

```
plugins:giop:message_server_binding_list=[ "BiDir_GIOP" , "GIOP" ];
```

Further information

For information on other variables used to set bidirectional GIOP, see [“policies:giop” on page 162](#). For details of all the steps involved in setting bidirectional GIOP, see the *Orbix Administrator's Guide*.

plugins:giop_snoop

The variables in this namespace configure settings for the GIOP Snoop tool. This tool intercepts and displays GIOP message content. Its primary roles are as a protocol-level monitor and a debug aid.

The GIOP Snoop plug-in implements message-level interceptors that can participate in client and/or server side bindings over any GIOP-based transport.

The variables in the `giop_snoop` namespace include the following:

- `ClassName`
- `filename`
- `rolling_file`
- `shlib_name`
- `verbosity`

ClassName

(Java only) `plugins:giop_snoop:ClassName` locates and loads the `giop_snoop` plug-in. The required classname is as follows:

```
plugins:giop_snoop:ClassName =  
    "com.iona.corba.giop_snoop.GIOPsnoopPlugIn";
```

To use the Java version of the GIOP Snoop plug-in, add the `giop_snoop.jar` file to your classpath. For example:

UNIX

```
export CLASSPATH=  
    $CLASSPATH:$IT_PRODUCT_DIR/asp/6.0/lib/asp-corba.jar
```

Windows

```
set CLASSPATH=
    %CLASSPATH%;%IT_PRODUCT_DIR%\asp\6.0\lib\asp-corba.jar
```

In addition, for both client or server configuration, the `giop_snoop` plug-in must be included in your `orb_plugins` list.

filename

`plugins:giop_snoop:filename` specifies a file for GIOP Snoop output. By default, output is directed to standard error (`stderr`). This variable has the following format:

```
plugins:giop_snoop:filename = "<some-file-path>";
```

A *month/day/year* time stamp is included in the output filename with the following general format:

```
<filename>.MMDDYYYY
```

rolling_file

`plugins:giop_snoop:rolling_file` prevents the GIOP Snoop output file from growing indefinitely. This setting specifies to open and then close the output file for each snoop message trace, instead of holding the output files open. This enables administrators to control the size and content of output files. This setting is enabled with:

```
plugins:giop_snoop:rolling_file = "true";
```

shlib_name

(C++ only) `plugins:giop_snoop:shlib_name` locates and loads the `giop_snoop` plug-in. This is configured by default as follows:

```
plugins:giop_snoop:shlib_name = "it_giop_snoop";
```

Note: In addition, for both client or server configuration, the `giop_snoop` plug-in must be included in your `orb_plugins` list.

verbosity

`plugins:giop_snoop:verbosity` is used to control the verbosity levels of the GIOP Snoop output. For example:

```
plugins:giop_snoop:verbosity = "1";
```

GIOP Snoop verbosity levels are as follows:

- | | |
|---|-----------|
| 1 | LOW |
| 2 | MEDIUM |
| 3 | HIGH |
| 4 | VERY HIGH |

plugins:http(s)

The variables in this namespace configure the http transport.

This namespace contains the following variables:

- `connection:max_unsent_data`
- `incoming_connections:hard_limit`
- `incoming_connections:soft_limit`
- `ip:send_buffer_size`
- `ip:receive_buffer_size`
- `outgoing_connections:hard_limit`
- `outgoing_connections:soft_limit`
- `pool:java_max_threads`
- `pool:java_min_threads`
- `pool:max_threads`
- `pool:min_threads`
- `tcp_connection:keep_alive`
- `tcp_connection:no_delay`
- `tcp_connection:linger_on_close`
- `tcp_listener:reincarnate_attempts`

connection:max_unsent_data

`connection:max_unsent_data` specifies, in bytes, the upper limit for the amount of unsent data associated with an individual connection. Defaults to 512Kb.

incoming_connections:hard_limit

`incoming_connections:hard_limit` specifies the maximum number of incoming (server-side) connections permitted to HTTP. HTTP does not accept new connections above this limit. Defaults to -1 (disabled).

incoming_connections:soft_limit

`incoming_connections:soft_limit` sets the number of connections at which HTTP begins closing incoming (server-side) connections. Defaults to -1 (disabled).

ip:send_buffer_size

`ip:send_buffer_size` specifies the `SO_SNDBUF` socket options to control how the IP stack adjusts the size of the output buffer. Defaults to 0, meaning the that buffer size is static.

ip:receive_buffer_size

`ip:receive_buffer_size` specifies the `SO_RCVBUF` socket options to control how the IP stack adjusts the size of the input buffer. Defaults to 0, meaning the that buffer size is static.

outgoing_connections:hard_limit

`outgoing_connections:hard_limit` sets the maximum number of outgoing (client-side) connections permitted to HTTP. HTTP does not allow new outgoing connections above this limit. Defaults to -1 (disabled).

outgoing_connections:soft_limit

`outgoing_connections:soft_limit` specifies the number of connections at which HTTP begins closing outgoing (client-side) connections. Defaults to -1 (disabled).

pool:java_max_threads

`pool:java_max_threads` specifies the maximum number of threads reserved from the `WorkQueue` to support tasks working on behalf of the Java ATLI transport. Defaults to 512.

pool:java_min_threads

`pool:java_min_threads` specifies the minimum number of threads reserved from the `WorkQueue` to support tasks working on behalf of the Java ATLI transport. Defaults to 10.

pool:max_threads

`pool:max_threads` specifies the maximum number of threads reserved from the `WorkQueue` to support tasks working on behalf of the ATLI transport. Defaults to 5.

pool:min_threads

`pool:min_threads` specifies the minimum number of threads reserved from the `WorkQueue` to support tasks working on behalf of the ATLI transport. Defaults to 1.

tcp_connection:keep_alive

`tcp_connection:keep_alive` specifies the setting of `SO_KEEPALIVE` on sockets used to maintain HTTP connections. If set to `TRUE`, the socket will send a *'keepalive probe'* to the remote host if the connection has been idle for a preset period of time. The remote system, if it is still running, will send an `ACK` response. Defaults to `TRUE`.

tcp_connection:no_delay

`tcp_connection:no_delay` specifies if `TCP_NODELAY` is set on the sockets used to maintain HTTP connections. If set to `false`, small data packets are collected and sent as a group. The algorithm used allows for no more than a 0.2 msec delay between collected packets. Defaults to `TRUE`.

tcp_connection:linger_on_close

`tcp_connection:linger_on_close` specifies the setting of `SO_LINGER` on all tcp connections to ensure that tcp buffers get cleared once a socket is closed. Defaults to `TRUE`.

tcp_listener:reincarnate_attempts

(Windows only)

`tcp_listener:reincarnate_attempts` specifies the number of times that a `Listener` recreates its listener socket after receiving a `SocketException`.

Sometimes a network error may occur, which results in a listening socket being closed. On Windows, you can configure the listener to attempt a reincarnation, which enables new connections to be established.

`reincarnate_attempts` only affects Java and C++ applications on Windows. Defaults to 0 (no attempts).

plugins:i18n

The variables in this namespace specify the codesets used to support international locales in JSPs and servlets.

The following variables are contained in this namespace:

- [characterencoding:iancharset-javaconverter-map](#)
- [characterencoding:url-inputcharset-map](#)
- [locale:locale-iancharset-map](#)

characterencoding:iancharset-javaconverter-map

`characterencoding:iancharset-javaconverter-map` specifies the mapping from an IANA character set to a corresponding Java converter. The entries are specified as follows:

```
plugins:i18n:characterencoding:iancharset-javaconverter-map=[ "i
ana-charset1=java-converter1", ... ];
```

characterencoding:url-inputcharset-map

`characterencoding:url-inputcharset-map` specifies the mapping from a JSP/servlet URL to a fallback encoding to use when handling `HttpServletRequest` parameters to the JSP/Servlet. Encodings specified by the JSP/servlet using `HttpServletRequest::setCharacterEncoding()` or `HttpServletRequest::setContentType()` take precedence. The entries are specified as follows:

```
plugins:i18n:characterencoding:url-inputcharset-map=[ "url1/*=cod
eset1", ... ];
```

locale:locale-iancharset-map

locale:locale-iancharset-map specifies the mapping from a locale to a codeset that makes sense for that locale. For example, the locale `kr_KO` could be mapped to the codeset `EUCK-KR`.

If a JSP or a servlet makes a `HttpServletResponse::setLocale(locale)` call, then the encoding associated with the specified locale will be used to encode any string parameters in the `HttpServletResponse`.

The entries are specified as follows:

```
plugins:ii18n:locale:locale-iancharset-map=[" locale1=codeset1",  
...];
```

plugins:iiop

The variables in this namespace configure active connection management, IIOp buffer management. For more information about active connection management, see the *Orbix Administrator's Guide*.

This namespace contains the following variables:

- `buffer_pools:recycle_segments`
- `buffer_pools:segment_preallocation`
- `connection:max_unsent_data`
- `incoming_connections:hard_limit`
- `incoming_connections:soft_limit`
- `ip:send_buffer_size`
- `ip:receive_buffer_size`
- `ip:reuse_addr`
- `outgoing_connections:hard_limit`
- `outgoing_connections:soft_limit`
- `pool:java_max_threads`
- `pool:java_min_threads`
- `pool:max_threads`
- `pool:min_threads`
- `tcp_connection:keep_alive`
- `tcp_connection:no_delay`
- `tcp_connection:linger_on_close`
- `tcp_listener:reincarnate_attempts`
- `tcp_listener:reincarnation_retry_backoff_ratio`
- `tcp_listener:reincarnation_retry_delay`

buffer_pools:recycle_segments

`plugins:iiop:buffer_pools:recycle_segments` specifies whether the recycling of IIOp buffer segments is enabled for Java applications. This reduces the amount of memory used by the ORB. Defaults to `true`.

buffer_pools:segment_preallocation

`plugins:iiop:buffer_pools:segment_preallocation` specifies the number of IIOp buffer segments to pre-allocate for Java applications. Defaults to 20.

connection:max_unsent_data

`plugins:iiop:connection:max_unsent_data` specifies the upper limit for the amount of unsent data associated with an individual connection. Defaults to 512k.

incoming_connections:hard_limit

`plugins:iiop:incoming_connections:hard_limit` specifies the maximum number of incoming (server-side) connections permitted to IIOp. IIOp does not accept new connections above this limit. Defaults to -1 (disabled).

incoming_connections:soft_limit

`plugins:iiop:incoming_connections:soft_limit` sets the number of connections at which IIOp begins closing incoming (server-side) connections. Defaults to -1 (disabled).

ip:send_buffer_size

`plugins:iiop:ip:send_buffer_size` specifies the `SO_SNDBUF` socket options to control how the IP stack adjusts the size of the output buffer. Defaults to 0, meaning the that buffer size is static.

ip:receive_buffer_size

`plugins:iiop:ip:receive_buffer_size` specifies the `SO_RCVBUF` socket options to control how the IP stack adjusts the size of the input buffer. Defaults to 0, meaning the that buffer size is static.

ip:reuse_addr

`plugins:iiop:ip:reuse_addr` specifies whether a process can be launched on an already used port. The default is `true`. Setting this to `false` switches `SO_REUSEADDR` to `false`. This does not allow a process to listen on the same port. An exception indicating that the address is already in use will be thrown.

outgoing_connections:hard_limit

`plugins:iiop:outgoing_connections:hard_limit` sets the maximum number of outgoing (client-side) connections permitted to IIOP. IIOP does not allow new outgoing connections above this limit. Defaults to -1 (disabled).

outgoing_connections:soft_limit

`plugins:iiop:outgoing_connections:soft_limit` specifies the number of connections at which IIOP begins closing outgoing (client-side) connections. Defaults to -1 (disabled).

pool:java_max_threads

`plugins:iiop:pool:java_max_threads` specifies the maximum number of threads reserved from the `WorkQueue` to support tasks working on behalf of the Java ATLI transport. Defaults to 512.

pool:java_min_threads

`plugins:iiop:pool:java_min_threads` specifies the minimum number of threads reserved from the `WorkQueue` to support tasks working on behalf of the Java ATLI transport. Defaults to 10.

pool:max_threads

`plugins:iiop:pool:max_threads` specifies the maximum number of threads reserved from the `WorkQueue` to support tasks working on behalf of the ATLI transport. Defaults to 5.

pool:min_threads

`plugins:iiop:pool:min_threads` specifies the minimum number of threads reserved from the `WorkQueue` to support tasks working on behalf of the ATLI transport. Defaults to 1.

tcp_connection:keep_alive

`plugins:iiop:tcp_connection:keep_alive` specifies the setting of `SO_KEEPALIVE` on sockets used to maintain IIOP connections. If set to `TRUE`, the socket will send a *'keepalive probe'* to the remote host if the connection has been idle for a preset period of time. The remote system, if it is still running, will send an `ACK` response. Defaults to `TRUE`.

tcp_connection:no_delay

`plugins:iiop:tcp_connection:no_delay` specifies if `TCP_NODELAY` is set on the sockets used to maintain IIOP connections. If set to `false`, small data packets are collected and sent as a group. The algorithm used allows for no more than a 0.2 msec delay between collected packets. Defaults to `TRUE`.

tcp_connection:linger_on_close

`plugins:iiop:tcp_connection:linger_on_close` specifies the setting of `SO_LINGER` on all tcp connections to ensure that tcp buffers get cleared once a socket is closed. Defaults to `TRUE`.

tcp_listener:reincarnate_attempts

(Windows only)

`tcp_listener:reincarnate_attempts` specifies the number of times that a `Listener` recreates its listener socket after receiving a `SocketException`.

Sometimes an network error may occur, which results in a listening socket being closed. On Windows, you can configure the listener to attempt a reincarnation, which enables new connections to be established. This variable only affects Java and C++ applications on Windows. Defaults to 0 (no attempts).

tcp_listener:reincarnation_retry_backoff_ratio

(Windows only)

`plugins:iiop:tcp_listener:reincarnation_retry_backoff_ratio` specifies the degree to which delays between retries increase from one retry to the next. Datatype is `long`. Defaults to 1. This variable only affects Java and C++ applications on Windows.

tcp_listener:reincarnation_retry_delay

(Windows only)

`plugins:iiop:tcp_listener:reincarnation_retry_delay` specifies a delay between reincarnation attempts. Data type is `long`. Defaults to 0 (no delay). This variable only affects Java and C++ applications on Windows.

plugins:ifr

The variables in this namespace control the persistence model of the interface repository. The interface repository can run in indirect persistent mode where it is accessed using the locator and node daemons. The interface repository can also run in direct persistent mode where it listens on a specified port number for requests.

This namespace contains the following variables:

- `advertise_services`
- `direct_persistence`
- `iiop:port`
- `iiop:host`

advertise_services

`advertise_services` specifies whether the ifr service should register plain text keys for the object references it publishes in prepare mode. Defaults to `true`.

direct_persistence

`direct_persistence` specifies if the interface repository runs in direct persistent mode. Defaults to `false` meaning that the service runs in indirect persistent mode. If it is set to `true`, the interface repository runs in direct persistent mode and the user must configure a port on which it will listen.

iiop:port

`iiop:port` specifies the port on which the interface repository listens when it is running in direct persistent mode. Only required when `direct_persistence` is set to `true`.

iiop:host

`iiop:host` specifies the host on which the interface repository is running. Only required when `direct_persistence` is set to `true`.

plugins:it_http_sessions

This namespace includes the following:

- [ClassName](#)

ClassName

`ClassName` specifies the default implementation which relies on cookies been accepted by the browser. The default implementation is enabled by specifying the plugin class name in the `orb_plugins` and `binding:servlet_binding_list`. For example:

```
plugins:it_http_sessions:ClassName="com.iona.servlet.session.HttpSessionPlugIn";
```

plugins:it_mgmt

This namespace includes the following:

- `managed_server_id:name`

managed_server_id:name

`managed_server_id:name` specifies the server name that you wish to appear in the IONA Administrator management console.

To enable management on a server, you must ensure that the following configuration variables are set:

```
plugins:orb:is_managed = true;
plugins:it_mgmt:managed_server_id:name = <your_server_name>;
```

plugins:it_mbean_monitoring

This namespace includes the following:

- `workqueue`.
- `sampling_period`.

workqueue

`plugins:it_mbean_monitoring:workqueue` specifies whether to enable monitoring of the ORB work queue MBean. Defaults to `false`. The ORB work queue is used to control the flow of requests. To enable work queue monitoring, set this variable as follows:

```
plugins:it_mbean_monitoring:workqueue = "true";
```

sampling_period

`plugins:it_mbean_monitoring:sampling_period` specifies the sampling interval for monitored MBean attributes. The default period is 100 milliseconds:

```
plugins:it_mbean_monitoring:sampling_period = "100";
```

plugins:it_pluggable_http_sessions

This namespace includes the following:

- [ClassName](#)
- [contexts](#)
- [mechanisms](#)
- [default_mechanism](#)

ClassName

`ClassName` specifies the classname for pluggable sessions. Pluggable sessions can be used instead of `it_http_sessions` (the default). Pluggable sessions allow custom session implementations and URL-encoding for session information.

To use the pluggable sessions, replace the `it_http_sessions` in the `orb_plugins` and `binding:servlet_binding_list` with `it_pluggable_http_sessions`. For example:

```
plugins:it_pluggable_http_sessions:ClassName="com.iona.servlet.s  
ession.PluggableHttpSessionPlugIn";
```

contexts

`contexts` specifies alternative session implementations to use per context root. The class name must implement the `com.iona.servlet.session.ExtendedHttpSessionFactory` interface. For example:

```
plugins:it_pluggable_http_sessions:contexts=["/myCtxRoot=myExten  
dedHttpSessionFactory",  
"/myAltRoot=myExtAltHttpSessionFactory"];
```

mechanisms

`mechanisms` specifies the mechanism used for passing session information to the client. This is also specified per context root. Possible values are:

- `url_rewriting` – URL rewriting is used.
- `cookies` – cookies are used.
- `mixed` – if the client supports `cookies`, these are used, otherwise `url_rewriting` is used.

For example:

```
plugins:it_pluggable_http_sessions:mechanisms=["/myCtxRoot=url_rewriting", "/myAltRoot=mixed"];
```

default_mechanism

`default_mechanism` specifies the mechanism for context roots not listed in the `mechanism` setting. If the `default_mechanism` setting is omitted, `cookies` are used as the default.

For example:

```
plugins:it_pluggable_http_sessions:default_mechanism="cookies";
```

plugins:it_response_time_collector

The variables in this namespace control the response time collector plugin. This is a performance logging plugin that is used to integrate Orbix with Enterprise Management Systems, such as IBM Tivoli. The collector plugin periodically harvests data from the response time logger and request counter plugins and logs the results.

The `it_response_time_collector` variables include the following:

- `period`
- `filename`
- `system_logging_enabled`
- `syslog_appID`
- `server-id`

period

`period` specifies the response time period. If you not specify a response time, this defaults to 60 seconds. For example:

```
plugins:it_response_time_collector:period = "90";
```

filename

`filename` specifies the filename used to log performance data. For example:

```
plugins:it_response_time_collector:filename =  
"/var/log/my_app/perf_logs/treasury_app.log";
```

system_logging_enabled

`system_logging_enabled` specifies if the collector logs to a syslog daemon or Windows event log. Values are `true` or `false`.

```
plugins:it_response_time_collector:system_logging_enabled =  
    "true";
```

syslog_appID

`syslog_appID` specifies an application name that is prepended to all syslog messages, for example:

```
plugins:it_response_time_collector:syslog_appID = "treasury";
```

If you do not specify an ID, the default is `iona`.

server-id

`server-id` specifies a server ID that will be reported in your log messages. This server ID is particularly useful in the case where the server is a replica that forms part of a cluster. In a cluster, the server ID enables management tools to recognize log messages from different replica instances. You can configure a server ID as follows:

```
plugins:it_response_time_collector:server-id = "Locator-1";
```

This setting is optional; and if omitted, the server ID defaults to the ORB name of the server. In a cluster, each replica must have this value set to a unique value to enable sensible analysis of the generated performance logs.

plugins:it_security_service

This namespace includes the following:

- [domain_list](#)
- [HOSTNAME](#)
- [init_at_startup_list](#)
- [default_domain](#)

domain_list

`domain_list` specifies the realms in this domain. The default values are ["DEFAULT", "FILE"].

HOSTNAME

`HOSTNAME` specifies the name of the class which implements user defined realms. For example:

```
plugins:it_security_service:domain_classname:DEFAULT="com.ionas.j2ee.security.securitydomains.DefaultSecurityDomainImpl";
```

init_at_startup_list

`init_at_startup_list` specifies the realms which are initialized at startup. The default values are ["DEFAULT", "FILE"].

default_domain

`default_domain` specifies the default realm if one is not selected in the web.xml file. The default value is "DEFAULT".

plugins:file_security_domain

This namespace includes the following:

- `file_list`
- `file_name`

file_list

`file_list` specifies the list of files for FILE (technical) realm. The default value is ["ASP"].

file_name

`file_name` specifies the location of the file, for example:

```
plugins:file_security_domain:file_name:ASP="%{IT_PRODUCT_DIR}/etc/security/SecurityDomain.xml";
```

plugins:jta

The variables in this namespace configure Java Transaction API plugin. It contains following configuration variables:

- `poa_namespace`
- `resource_poa_name`
- `enable_recovery`

poa_namespace

`poa_namespace` specifies the name of the transient POA namespace used for persistent POA objects. Defaults to `iJTA`.

resource_poa_name

`resource_poa_name` specifies the name of the persistent POA used by recoverable JTA objects. Defaults to `resource`.

enable_recovery

`enable_recovery` is a boolean which specifies whether the JTA is capable of recovery. This must be set to `true` when JTA is used in conjunction with a 2PC transaction manager. Defaults to `false`.

kdm_enabled

`kdm_enabled` specifies if the KDM server plugin is enabled. When equal to `true`, the KDM server plugin is enabled; when equal to `false`, the KDM server plugin is disabled. Default is `true`.

iiop_tls:port

`iiop_tls:port` specifies the well known IP port on which the KDM server listens for incoming calls.

checksums_optional

`checksums_optional` specifies if the secure information associated with a server is required to include a checksum. When equal to `false`, the secure information associated with a server must include a checksum; when equal to `true`, the presence of a checksum is optional. Default is `false`.

plugins:local_log_stream

The variables in this namespace configure how Orbix logs runtime information. By default, Orbix is configured to log messages to standard error. You can change this behavior for an ORB by specifying the `local_log_stream` plug-in. This namespace contains the following variables:

- `buffer_file`
- `filename`
- `log_elements`
- `milliseconds_to_log`
- `rolling_file`

For full details of Orbix logging, see the *Orbix Administrator's Guide*.

buffer_file

`buffer_file` specifies whether the output stream is buffered. This is expressed as a boolean value. The default is `false`. To enable buffer file behavior, set this variable to `true`. For example:

```
plugins:local_log_stream:buffer_file = "true";
```

When this is set to `true`, by default, the local log stream is output to file every 1000 milliseconds when there are more than 100 log messages in the buffer. You can change this behavior by updating the `log_elements` and `milliseconds_to_log` variables.

filename

`filename` sets the output stream to the specified local file. For example:

```
plugins:local_log_stream:filename = "/var/adm/mylocal.log";
```

log_elements

`log_elements` specifies the minimum number of log messages in the buffer before each output to a file. This is expressed as an integer value. The default is 100. You can update this value to suit your environment. For example:

```
plugins:local_log_stream:log_elements = "200";
```

milliseconds_to_log

`milliseconds_to_log` specifies the time interval between each output to a file. This is expressed as an integer value. The default is 1000. You can update this value to suit your environment. For example:

```
plugins:local_log_stream:milliseconds_to_log = "2000";
```

rolling_file

`rolling_file` is a boolean which specifies that the logging plugin is to use a rolling file to prevent the local log from growing indefinitely. In this model, the stream appends the current date to the configured filename. This produces a complete filename—for example:

```
/var/adm/art.log.02171999
```

A new file begins with the first event of the day and ends at 23:59:59 each day.

The default behavior is `true`. To disable rolling file behavior, set this variable to `false`. For example:

```
plugins:local_log_stream:rolling_file = "false";
```

plugins:locator

The variables in this namespace configure the locator daemon plugin. The locator daemon enables clients to locate servers in a network environment.

This namespace contains the following variables:

- `allow_node_daemon_change`
- `iop:port`
- `iop_tls:port`
- `location_domain_name`
- `node_daemon_heartbeat_interval`
- `nt_service_dependencies`
- `refresh_master_interval`

allow_node_daemon_change

`allow_node_daemon_change` specifies whether is it possible to start a process under a different node daemon than the node daemon it was originally registered with.

This is only applicable to processes that are not already active and are not registered to be launched on demand. This enables you to move a process to another node without performing any administration actions. You can move a process to a new host by stopping it on its current host, and restarting it on the new host. The default is `true`.

iiop:port

`iiop:port` specifies the IIOP (Internet Inter-ORB Protocol) port for the locator daemon.

iiop_tls:port

`iiop_tls:port` specifies the IIOP/TLS port for the locator daemon. For information on configuring security, see the *Security Guide*.

Note: This is only useful for applications that have a single TLS listener. For applications that have multiple TLS listeners, you need to programmatically specify the well-known addressing policy.

location_domain_name

`location_domain_name` sets the name of the currently configured location domain. Defaults to `Default Location Domain`.

node_daemon_heartbeat_interval

`node_daemon_heartbeat_interval` specifies, in seconds, the interval between heartbeat messages sent by the locator to its node daemons. This is used to detect the failure of a node daemon. The default interval is 30 seconds. See also [heartbeat_interval_timeout](#).

nt_service_dependencies

`nt_service_dependencies` list the locator daemon's dependencies on other NT services. The dependencies are listed in the following format:

```
IT ORB-name domain-name
```

This variable only has meaning if the locator daemon is installed as an NT service.

refresh_master_interval

`refresh_master_interval` specifies the maximum number of seconds that a slave locator replica waits for a new master to be declared.

A new master is declared after a failed attempt to delegate an operation to the current master. If no master is found during the specified interval of time, a `TRANSIENT` exception is raised. Defaults to 60.

For example:

```
plugins:locator:refresh_master_interval="40";
```

plugins:naming

The variables in this namespace configure the naming service plugin. The naming service allows you to associate abstract names with CORBA objects, enabling clients to locate your objects.

This namespace contains the following variables:

- `advertise_services`
- `destructive_methods_allowed`
- `direct_persistence`
- `iiop:port`
- `lb_default_initial_load`
- `lb_default_load_timeout`
- `max_tx_retries`
- `nt_service_dependencies`
- `refresh_master_interval`

advertise_services

`advertise_services` specifies whether the naming service should register plain text keys for the object references it publishes in prepare mode. Defaults to `true`.

destructive_methods_allowed

`destructive_methods_allowed` specifies if users can make destructive calls, such as `destroy()`, on naming service elements. The default value is `true`, meaning the destructive methods are allowed.

direct_persistence

`direct_persistence` specifies if the service runs using direct or indirect persistence. The default value is `false`, meaning indirect persistence.

iiop:port

`iiop:port` specifies the port that the service listens on when running using direct persistence.

lb_default_initial_load

`lb_default_initial_load` specifies the default initial load value for a member of an active object group. The load value is valid for a period of time specified by the timeout assigned to that member. Defaults to 0.0. For more information, see the *Orbix Administrator's Guide*.

lb_default_load_timeout

`lb_default_load_timeout` specifies the default load timeout value for a member of an active object group. The default value of -1 indicates no timeout. This means that the load value does not expire. For more information, see the *Orbix Administrator's Guide*.

max_tx_retries

`max_tx_retries` specifies the maximum number of times that certain transactions are retried in the event of a failure. This currently only applies to transactions that run during the initialization of a slave. Defaults to 3.

nt_service_dependencies

`nt_service_dependencies` specifies the naming service's dependencies on other NT services. The dependencies are listed in the following format:

```
IT ORB-name domain-name
```

This variable only has meaning if the naming service is installed as an NT service.

refresh_master_interval

`refresh_master_interval` specifies the maximum number of seconds that a slave naming service replica waits for a new master to be declared.

A new master is declared after a failed attempt to delegate an operation to the current master. If no master is found during the specified interval of time, a `TRANSIENT` exception is raised. Defaults to 60.

For example:

```
plugins:naming:refresh_master_interval = 40;
```

plugins:node_daemon

The variables in this namespace configure the node daemon plugin. The node daemon, in conjunction with the location daemon, enables on-demand activation of servers in a network environment.

This namespace contains the following variables:

- `heartbeat_interval_timeout`
- `is_managed`
- `iiop:port`
- `iiop_tls:port`
- `recover_processes`
- `register_interval`

heartbeat_interval_timeout

`heartbeat_interval_timeout` specifies, in seconds, the interval a node daemon expects to receive a heartbeat message from a locator.

If no heartbeat is received in this interval the node daemon attempts to register with the locator again. The default is 40 seconds.

See also [node_daemon_heartbeat_interval](#).

is_managed

`is_managed` specifies whether or not the node daemon is managed using the management service. Defaults to `false`.

iiop:port

`iiop:port` specifies the Internet Inter-ORB Protocol (IIOP) port on which the node daemon listens.

iiop_tls:port

`iiop_tls:port` specifies the Internet Inter-ORB Protocol/Transport Layer Security (IIOP/TLS) port on which the node daemon listens. For information on configuring security, see the *Security Guide*.

recover_processes

`recover_processes` specifies the behavior of the node daemon at startup. By default, when starting up, the node daemon attempts to contact the CORBA servers that it was managing during its previous run.

To speed up the time required to start up when managing large numbers of CORBA servers, you can set the `recover_process` environment variable as follows:

```
plugins:node_daemon:recover_processes=false
```

register_interval

`register_interval` specifies, in seconds, the interval between attempts by a node daemon to register with its locators. This occurs at startup if a locator is not available or if a locator has not sent a heartbeat message in the time interval specified by the variable `heartbeat_interval_timeout`. The default interval is 5 seconds.

plugins:notify

The variables in this namespace configure the behavior of the notification service. It contains the following variables:

- `advertise_services`
- `dispatch_strategy`
- `dispatch_threads`
- `direct_persistence`
- `events_per_transaction`
- `event_queue`
- `iiop:port`
- `trace:database`
- `trace:events`
- `trace:filters`
- `trace:lifecycle`
- `trace:locks`
- `trace:queue`
- `trace:retry`
- `trace:subscription`
- `trace:transactions`

advertise_services

`advertise_services` specifies whether the notify service should register plain text keys for the object references it publishes in prepare mode. Defaults to true.

dispatch_strategy

`dispatch_strategy` specifies the method used for allocating threads to dispatch events.

You can set this variable to `single_thread` or `thread_pool`:

- `single_thread` (default) specifies that each proxy has its own thread for invoking requests on the client supplier or consumer. The application is responsible for managing its own threads. This setting requires that pull suppliers implement the `pull()` method.
 - `thread_pool` specifies that the notification service allocates threads for each consumer request, and manages the thread pool. The number of available threads is set by `dispatch_threads`. This setting requires that pull suppliers implement the `try_pull()` method.
-

dispatch_threads

`dispatch_threads` specifies the number of threads available to dispatch events, if `dispatch_strategy` is set to `thread_pool`. The default is 10.

direct_persistence

`direct_persistence` specifies if the notification service runs using direct or indirect persistence. The default value is `FALSE`, meaning indirect persistence. If you set the value to `TRUE`, you must also set `iiop:port`.

events_per_transaction

`events_per_transaction` specifies the number of events selected per database transaction for transmission to a push consumer. This variable reduces the total transmission overhead for persistent events. The default value is 10.

event_queue

`event_queue` specifies whether the notification channel holds events in a queue before dispatching them or dispatches events as they come in.

You can set this variable to `true` or `false`:

- `true` tells the channel to use a messaging queue. This can improve performance for applications with a large number of events passing through the channel.

- `false` (default) tells the channel to dispatch events as they are received.
-

iiop:port

`iiop:port` specifies the port that the service listens on when using direct persistence.

trace:database

`trace:database` specifies the amount of diagnostic information to record about the behavior of the service's persistent database. Set this value to 1 or greater to enable tracing. The default is 0 (no logging).

trace:events

`trace:events` specifies the amount of diagnostic information logged about events passing through the notification channel. Set this value to 1 or greater to enable tracing. The default is 0 (no logging).

trace:filters

`trace:filters` specifies the amount of information logged by filters in the notification channel. The default is 0.

trace:lifecycle

`trace:lifecycle` specifies the amount of diagnostic information logged about service object (channel, admin, proxy) lifecycles. The default is 0.

trace:locks

`trace:locks` specifies the amount of diagnostic information logged about locks on the service's persistent database. The default is 0.

trace:queue

`trace:queue` specifies the amount of information logged about the notification service's event queue. The default is 0.

trace:retry

`trace:retry` specifies the amount of diagnostic information logged about retried event transmissions. The default is 0.

trace:subscription

`trace:subscription` specifies the amount of information logged about clients publishing and subscribing to events. The default is 0.

trace:transactions

`trace:transactions` specifies the amount of information logged about transactions with the service's persistent database. The default is 0.

plugins:notify:database

The variables in this namespace control the behavior of the notification service's database. It contains the following variables:

- `checkpoint_archive_old_files`
- `checkpoint_deletes_old_logs`
- `checkpoint_interval`
- `checkpoint_min_size`
- `data_dir`
- `db_home`
- `log_dir`
- `lk_max`
- `max_retries`
- `max_sleep_time`
- `tx_max`
- `mode`
- `old_log_dir`
- `private`
- `recover_fatal`
- `sync_transactions`
- `tmp_dir`

checkpoint_archive_old_files

`checkpoint_archive_old_files` specifies whether the notification service retains archives of the old logs after each checkpoint. When this property is set to `true`, old logs are moved to `old_log_dir`. Defaults to `false`.

checkpoint_deletes_old_logs

`checkpoint_deletes_old_logs` specifies whether the notification service deletes old log files for its database after each checkpoint. Defaults to `true`.

checkpoint_interval

`checkpoint_interval` specifies, in seconds, the checkpoint interval for posting data from the transaction log file to the notification service's database. To disable checkpointing, set this variable to 0. The default is 300.

checkpoint_min_size

`checkpoint_min_size` specifies the amount of data, in kilobytes, to checkpoint at a time. The default is 65536.

data_dir

`data_dir` specifies the directory where the data files are stored; relative paths are relative to `db_home`. The directory must be on a local file system. Defaults to `data`.

db_home

`db_home` must point to the home directory of the Berkeley DB database.

log_dir

`log_dir` specifies the directory where the log files are stored; relative paths are relative to `db_home`. The directory must be on a local file system. For maximum performance and reliability, place data files and log files on separate disks, managed by different disk controllers. Defaults to `logs`.

lk_max

`lk_max` specifies the maximum number of locks allowed on the database at a time. The default is 16384.

max_retries

`max_retries` specifies the maximum number of times to retry database transactions before aborting. The default is 0 (infinite).

max_sleep_time

`max_sleep_time` specifies the maximum number of seconds to sleep while waiting for a database transaction to complete. The time between successive retries grows exponentially until this value is reached, that is 1, 2, 4, 8,... `max_sleep_time`. Setting this variable to 0 disables sleeping between retries. The default is 256.

tx_max

`tx_max` specifies the maximum number of concurrent database transactions allowed at any one time. This property should be set proportional to the number of persistent proxies. If the number of persistent proxies outpaces the number of transactions allowed, performance will degrade. The default is 0 (infinite).

mode

`mode` specifies the file mode on UNIX platforms. Defaults to 0.

old_log_dir

`old_log_dir` specifies the directory into which old transaction log files are moved if `checkpoint_deletes_old_logs` is set to `false`. Defaults to `old_logs`.

private

`private` specifies whether only one process is permitted to use this environment. Set to `false` when you want to obtain statistics on your database with `db_stat`. Defaults to `true`.

recover_fatal

`recover_fatal` specifies whether to perform fatal recovery instead of normal recovery. Defaults to `false`.

sync_transactions

`sync_transactions` specifies whether to use synchronous or asynchronous database transactions.

You can set this variable to `true` or `false`:

- `true` (default) specifies using synchronous database transactions. The channel blocks until the transaction is complete.
 - `false` specifies using asynchronous database transactions. The channel issues the transaction and continues.
-

tmp_dir

`tmp_dir` specifies the directory for temporary files. The directory must be on a local file system. Defaults to `tmp`.

plugins:notify_log

The variables in this namespace control the behavior of notify log service. These variables include the following:

- `advertise_services`
- `is_managed`
- `shlib_name`

advertise_services

`advertise_services` specifies whether the notify_log service should register plain text keys for the object references it publishes in prepare mode. Defaults to `true`.

is_managed

`is_managed` specifies whether or not the notify log service can be managed using the management service. Defaults to `false`, which means the management service does not manage the service.

shlib_name

`shlib_name` identifies the shared library (or DLL in Windows) containing the plugin implementation. The notify log plugin is associated with the base name of the shared library (`it_notify_log_svr` in this case). This library base name is expanded in a platform-dependent manner to obtain the full name of the library file.

```
plugins:basic_log:shlib_name = "it_notify_log_svr";
```

plugins:orb

The `plugins:orb` namespace includes the `plugins:orb:is_managed` configuration variable.

is_managed

`is_managed` specifies whether or not the ORB can be managed using the management service. Defaults to `false`, which means the management service cannot manage the server ORB.

To enable management on a server, you must ensure that the following configuration variables are set:

```
plugins:orb:is_managed = true;
plugins:it_mgmt:managed_server_id:name = <your_server_name>;
```

Set `<your_server_name>` to whatever server name you want to appear in the IONA Administrator management console.

plugins:ots

The variables in this namespace configure the object transaction service (OTS) generic plugin. The generic OTS plugin contains client and server side transaction interceptors and the implementation of `CosTransactions::Current`. For details of this plugin, refer to the *CORBA OTS Guide*.

The `plugins:ots` namespace contains the following variables:

- `advertise_services`
- `concurrent_transaction_map_size`
- `default_ots_policy`
- `default_transaction_policy`
- `default_transaction_timeout`
- `interposition_style`
- `jit_transactions`
- `ots_vll_policy`
- `propagate_separate_tid_optimization`
- `rollback_only_on_system_ex`
- `support_ots_vll`
- `transaction_factory_name`

advertise_services

`advertise_services` specifies whether the ots service should register plain text keys for the object references it publishes in prepare mode. Defaults to true.

concurrent_transaction_map_size

`concurrent_transaction_map_size` specifies the initial size of a hash table used when dealing with concurrently propagated transactions. Defaults to 15. This variable only affects Java applications

default_ots_policy

`default_ots_policy` specifies the default `OTSPolicy` value used when creating a POA. Set to one of the following values:

`requires`
`forbids`
`adapts`

If no value is specified, no `OTSPolicy` is set for new POAs.

default_transaction_policy

`default_transaction_policy` specifies the default `TransactionPolicy` value used when creating a POA.

Set to one of the following values:

- `requires` corresponds to a `TransactionPolicy` value of `Requires_shared`.
- `allows` corresponds to a `TransactionPolicy` value of `Allows_shared`.

If no value is specified, no `TransactionPolicy` is set for new POAs.

default_transaction_timeout

`default_transaction_timeout` specifies the default timeout, in seconds, of a transaction created using `CosTransactions::Current`. A value of zero or less specifies no timeout. Defaults to 30 seconds.

interposition_style

`interposition_style` specifies the style of interposition used when a transaction first visits a server. Set to one of the following values:

- `standard`: A new subordinator transaction is created locally and a resource is registered with the superior coordinator. This subordinate transaction is then made available through the `Current` object.
- `proxy`: (default) A locally constrained proxy for the imported transaction is created and made available through the `Current` object.

Proxy interposition is more efficient, but if you need to further propagate the transaction explicitly (using the `Control` object), standard interposition must be specified.

jit_transactions

`jit_transactions` is a boolean which determines whether to use just-in-time transaction creation. If set to `true`, transactions created using `Current::begin()` are not actually created until necessary. This can be used in conjunction with an `OTSPolicy` value of `SERVER_SIDE` to delay creation of a transaction until an invocation is received in a server. Defaults to `false`.

ots_v11_policy

`ots_v11_policy` specifies the effective `OTSPolicy` value applied to objects determined to support `CosTransactions::TransactionalObject`, if `support_ots_v11` is set to `true`.

Set to one of the following values:

- `adapts`
 - `requires`
-

propagate_separate_tid_optimization

`propagate_separate_tid_optimization` specifies whether an optimization is applied to transaction propagation when using C++ applications. Must be set for both the sender and receiver to take affect. Defaults to `true`.

rollback_only_on_system_ex

`rollback_only_on_system_ex` specifies whether to mark a transaction for rollback if an invocation on a transactional object results in a system exception being raised. Defaults to `true`.

support_ots_v11

`support_ots_v11` specifies whether there is support for the OMG OTS v1.1 `CosTransactions::TransactionalObject` interface. This option can be used in conjunction with `ots_v11_policy`. When this option is enabled, the OTS interceptors might need to use remote `_is_a()` calls to determine the type of an interface. Defaults to `false`.

transaction_factory_name

`transaction_factory_name` specifies the initial reference for the transaction factory. This option must match the corresponding entry in the configuration scope of your transaction service implementation. Defaults to `TransactionFactory`.

plugins:ots_lite

The variables in this namespace configure the Lite implementation of the object transaction service. The `ots_lite` plugin contains an implementation of `CosTransactions::TransactionFactory` which is optimized for use in a single resource system. For details, see the *CORBA Programmer's Guide*.

This namespace contains the following variables:

- `orb_name`
- `otid_format_id`
- `superior_ping_timeout`
- `transaction_factory_name`
- `transaction_timeout_period`
- `use_internal_orb`

orb_name

`orb_name` specifies the ORB name used for the plugin's internal ORB when `use_internal_orb` is set to `true`. The ORB name determines where the ORB obtains its configuration information and is useful when the application ORB configuration needs to be different from that of the internal ORB. Defaults to the ORB name of the application ORB.

otid_format_id

`otid_format_id` specifies the value of the `formatID` field of a transaction's identifier (`CosTransactions::otid_t`). Defaults to `0x494f4e41`.

superior_ping_timeout

`superior_ping_timeout` specifies, in seconds, the timeout between queries of the transaction state, when standard interposition is being used to recreate a foreign transaction. The interposed resource periodically queries the recovery coordinator, to ensure that the transaction is still alive when the timeout of the superior transaction has expired. Defaults to 30.

transaction_factory_name

`transaction_factory_name` specifies the initial reference for the transaction factory. This option must match the corresponding entry in the configuration scope of your generic OTS plugin to allow it to successfully resolve a transaction factory. Defaults to `TransactionFactory`.

transaction_timeout_period

`transaction_timeout_period` specifies the time, in milliseconds, of which all transaction timeouts are multiples. A low value increases accuracy of transaction timeouts, but increases overhead. This value is added to all transaction timeouts. To disable all timeouts, set to 0 or a negative value. Defaults to 1000.

use_internal_orb

`use_internal_orb` specifies whether the `ots_lite` plugin creates an internal ORB for its own use. By default, `ots_lite` creates POAs in the application's ORB. This option is useful if you want to isolate the transaction service from your application ORB. Defaults to `false`.

plugins:ots_encina

The `plugins:ots_encina` namespace stores configuration variables for the Encina OTS plugin. The `ots_encina` plugin contains an implementation of IDL interface `CosTransactions::TransactionFactory` that supports the recoverable 2PC protocol. For details, see the *CORBA OTS Guide*.

This namespace contains the following variables:

- `agent_ior_file`
- `allow_registration_after_rollback_only`
- `backup_restart_file`
- `create_transaction_mbeans`
- `direct_persistence`
- `global_namespace_poa`
- `iiop:port`
- `initial_disk`
- `initial_disk_size`
- `log_threshold`
- `log_check_interval`
- `max_resource_failures`
- `namespace_poa`
- `orb_name`
- `otid_format_id`
- `resource_retry_timeout`
- `restart_file`
- `trace_comp`
- `trace_file`
- `trace_on`
- `transaction_factory_name`
- `transaction_factory_ns_name`
- `transaction_timeout_period`
- `use_internal_orb`
- `use_raw_disk`

agent_ior_file

`agent_ior_file` specifies the file path where the management agent object's IOR is written. Defaults to an empty string.

allow_registration_after_rollback_only

`allow_registration_after_rollback_only` (C++ only) specifies whether registration of resource objects is permitted after a transaction is marked for rollback.

- `true` specifies that resource objects can be registered after a transaction is marked for rollback.
- `false` (default) specifies that resource objects cannot be registered once a transaction is marked for rollback.

This has no effect on the outcome of the transaction.

backup_restart_file

`backup_restart_file` specifies the path for the backup restart file used by the Encina OTS to locate its transaction logs. If unspecified, the backup restart file is the name of the primary restart file—set with `restart_file`—with a `.bak` suffix. Defaults to an empty string.

create_transaction_mbeans

`create_transaction_mbeans` (Java only) specifies whether OTS management objects are created. Defaults to `true`.

direct_persistence

`direct_persistence` specifies whether the transaction factory object can use explicit addressing—for example, a fixed port. If set to `true`, the addressing information is picked up from `plugins:ots_encina`. For example, to use a fixed port, set `plugins_ots_encina:iiop:port`. Defaults to `false`.

global_namespace_poa

`global_namespace_poa` specifies the top-level transient POA used as a namespace for OTS implementations. Defaults to `iOTS`.

iiop:port

`iiop:port` specifies the port that the service listens on when using direct persistence.

initial_disk

`initial_disk` specifies the path for the initial file used by the Encina OTS for its transaction logs. Defaults to an empty string.

initial_disk_size

`initial_disk_size` specifies the size of the initial file used by the Encina OTS for its transaction logs. Defaults to `2`.

log_threshold

`log_threshold` specifies the percentage of transaction log space, which, when exceeded, results in a management event. Must be between `0` and `100`. Defaults to `90`.

log_check_interval

`log_check_interval` specifies the time, in seconds, between checks for transaction log growth. Defaults to 60.

max_resource_failures

`max_resource_failures` specifies the maximum number of failed invocations on `CosTransaction::Resource` objects to record. Defaults to 5.

namespace_poa

`namespace_poa` specifies the transient POA used as a namespace. This is useful when there are multiple instances of the plugin being used; each instance must use a different namespace POA to distinguish itself. Defaults to `Encina`.

orb_name

`orb_name` specifies the ORB name used for the plugin's internal ORB when `use_internal_orb` is set to `true`. The ORB name determines where the ORB obtains its configuration information, and is useful when the application ORB configuration needs to be different from that of the internal ORB. Defaults to the ORB name of the application ORB.

otid_format_id

`otid_format_id` specifies the value of the `formatID` field of a transaction's identifier (`CosTransactions::otid_t`). Defaults to `0x494f4e41`.

resource_retry_timeout

`resource_retry_timeout` specifies the time, in seconds, between retrying a failed invocation on a resource object. A negative value means the default is used. Defaults to 5.

restart_file

`restart_file` specifies the path for the restart file used by the Encina OTS to locate its transaction logs. Defaults to an empty string.

trace_comp

`trace_comp` sets the Encina trace levels for the component `comp`, where `comp` is one of the following:

```
bde
log
restart
tran
tranLog_log
tranLog_tran
util
vol
```

Set this variable to a bracket-enclosed list that includes one or more of the following string values:

- `event`: interesting events.
- `entry`: entry to a function.
- `param`: parameters to a function.
- `internal_entry`: entry to internal functions.
- `internal_param`: parameters to internal functions.
- `global`.

Defaults to `[]`.

trace_file

`trace_file` specifies the file to which Encina level tracing is written when enabled via `trace_on`. If not set or set to an empty string, Encina level transactions are written to standard error. Defaults to an empty string.

trace_on

`trace_on` specifies whether Encina level tracing is enabled. If set to `true`, the information that is output is determined from the trace levels (see [trace_comp](#)). Defaults to `false`.

transaction_factory_name

`transaction_factory_name` specifies the initial reference for the transaction factory. This option must match the corresponding entry in the configuration scope of your generic OTS plugin to allow it to successfully resolve a transaction factory. Defaults to `TransactionFactory`.

transaction_factory_ns_name

`transaction_factory_ns_name` specifies the name used to publish the transaction factory reference in the naming service. Defaults to an empty string.

transaction_timeout_period

`transaction_timeout_period` specifies the time, in milliseconds, of which all transaction timeouts are multiples. A low value increases accuracy of transaction timeouts, but increases overhead. This value multiplied to all transaction timeouts. To disable all timeouts, set to `0` or a negative value. Defaults to `1000`.

use_internal_orb

`use_internal_orb` specifies whether the `ots_encina` plugin creates an internal ORB for its own use. By default the `ots_encina` plugin creates POA's in the application's ORB. This option is useful if you want to isolate the transaction service from your application ORB. Defaults to `false`.

use_raw_disk

`use_raw_disk` specifies whether the path specified by `initial_disk` is of a raw disk (`true`) or a file (`false`). If set to `false` and the file does not exist, the Encina OTS plugin tries to create the file with the size specified in `initial_disk_size`. Defaults to `false`.

plugins:ots_mgmt

The variables in this namespace configure the OTS Lite management plugin. All configuration variables in this namespace are for Java only.

This namespace contains the following variables:

- `create_transaction_mbeans`
- `enabled`
- `jmx_httpd_enabled`
- `transaction_manager_name`
- `jmx_httpd_port`

create_transaction_mbeans

`create_transaction_mbeans` specifies whether to create OTS management objects. Default to `false`.

enabled

`enabled` specifies whether management is enabled. Defaults to `false` meaning management is disabled.

jmx_httpd_enabled

`jmx_httpd_enabled` specifies whether the OTS management objects are available via JMX over HTTP. Defaults to `false`.

transaction_manager_name

`transaction_manager_name` specifies the name of the OTS transaction manager. Defaults to `OTS Lite Transaction Manager`.

jmx_httpd_port

`jmx_httpd_port` specifies the HTTP port number used when [jmx_httpd_enabled](#) is set to true. Defaults to 8082.

plugins:poa

This namespace contains variables to configure the CORBA POA plugin. It contains the following variables:

- [ClassName](#)
- [root_name](#)

ClassName

`ClassName` specifies the Java class in which the `poa` plugin resides. This is specified as follows:

```
plugins:poa:ClassName = "com.ionacorba.poa.POAPugin";
```

root_name

`root_name` specifies the name of the root POA, which is added to all fully-qualified POA names generated by that POA. If this variable is not set, the POA treats the root as an anonymous root, effectively acting as the root of the location domain.

plugins:pss

For C++ applications, the `plugins:pss` namespace stores configuration variables for the Persistent State Service (PSS) plugin. PSS is a CORBA service for building CORBA servers that access persistent data.

The following variables are contained in this namespace:

- [disable_caching](#)

For more details of this service, refer to the *CORBA Programmer's Guide*.

disable_caching

`disable_caching` specifies whether caching is disabled. When set to `true`, PSS does not perform any caching. This is useful for testing, and causes core dumps in code that does not manage PSS objects correctly. Defaults to `false`.

plugins:pss_db:envs:*env-name*

For C++ applications, the `plugins:pss_db:envs:env-name` namespace contains variables for the Persistent State Service (PSS) database plugin, where *env-name* represents the environment name. For example, `it_locator` represents persistent storage for the locator daemon. For details about this service, refer to the *CORBA Programmer's Guide*.

The following variables are contained in this namespace:

- `allow_minority_master`
- `always_download`
- `cache_size_gbytes`
- `cache_size_bytes`
- `checkpoint_archives_old_logs`
- `checkpoint_deletes_old_logs`
- `checkpoint_min_size`
- `concurrent_users`
- `create_dirs`
- `data_dir`
- `db_home`
- `deadlock_detector_aborts`
- `election_backoff_ratio`
- `election_delay`
- `election_init_timeout`
- `init_rep`
- `init_txn`
- `lg_bsize`
- `lg_max`
- `lk_max_lockers`
- `lk_max_locks`
- `lk_max_objects`
- `log_dir`
- `log_stats`
- `old_log_dir`

- `master_heartbeat_interval`
- `max_buffered_msgs`
- `max_buffered_msgs_size`
- `max_elections`
- `max_log_recs`
- `max_rep_threads`
- `min_log_recs`
- `mp_mmapsize`
- `ncache`
- `prevent_unilateral_promotion`
- `private`
- `recover_fatal`
- `rep_limit`
- `replica_name`
- `replica_priority`
- `run_deadlock_detector`
- `tmp_dir`
- `tx_max`
- `verb_all`
- `verb_checkpoint`
- `verb_deadlock`
- `verb_recovery`
- `verb_replication`
- `verb_waitsfor`

allow_minority_master

`allow_minority_master` specifies whether a master replica can exist without a full majority of active replicas. To allow a master to exist with only a minority of running replicas, set this variable to `true`.

Setting this variable to `true` only takes effect if there are two replicas in the replication group. This enables the only slave replica to be promoted if the master fails. Defaults to `false`.

Note: Enabling a minority master should be performed with caution. For example, a network partition can cause a slave to be promoted when the master is still running, leading to a duplicate master. Also, after a slave has been promoted, the old master must not be restarted when the new master is not running because updates made after the promotion will be lost.

always_download

`always_download` specifies when a slave replica should download the database environment from the master. Setting this to `true` means that the database environment is always downloaded from the master each time the slave starts.

Setting this to `false` means the database environment is downloaded the first time the slave is initialized, or when the slave becomes too far outdated with respect to the master. Defaults to `false`.

cachesize_gbytes

`cachesize_gbytes` specifies the value of the `gbytes` parameter passed to the `set_cachesize()` Berkeley DB function. There is no default value.

For more details, see the Berkeley DB documentation, available from <http://www.sleepycat.com/>.

cache_size_bytes

`cache_size_bytes` specifies the value of the `bytes` parameter passed to the `set_cache_size()` Berkeley DB function. There is no default value. For more details, see the Berkeley DB documentation, available from <http://www.sleepycat.com/>.

checkpoint_period

`checkpoint_period` is used in TX mode only, and specifies the transaction log checkpoint period in minutes. Defaults to 15.

checkpoint_archives_old_logs

`checkpoint_archives_old_logs` specifies whether the PSS archives old log files in the `old_logs` directory. To archive old log files, set this variable to `true`. Defaults to `false`.

checkpoint_deletes_old_logs

`checkpoint_deletes_old_logs` is used in TX mode only, and specifies whether the PSS deletes old log files after each checkpoint. When `false`, the PSS moves old log files to the `old_logs` directory. Defaults to `true`.

checkpoint_min_size

`checkpoint_min_size` is used in TX mode only, and specifies the minimum checkpoint size. If less than the `checkpoint_min_size` of data is written to the log since the last checkpoint, do not checkpoint. Defaults to 0.

concurrent_users

`concurrent_users` specifies the number of threads expected to use this environment at the same time. Defaults to 20.

create_dirs

`create_dirs` specifies whether the `db_home`, `log` and `tmp` directories are to be created, if they do not exist. Defaults to `false`.

data_dir

`data_dirs` specifies the directory where the data files are stored; relative paths are relative to `db_home`. The directory must be on a local file system. Defaults to `data`.

db_home

`db_home` specifies the home directory of the Berkeley DB database. For example, `plugins:pss_db:envs:it_locator:db_home` specifies the home directory for the locator daemon.

deadlock_detector_aborts

`deadlock_detector_aborts` specifies when the deadlock detector aborts, when the value of `run_deadlock_detector` is set to `true`. Set this variable to one of the following:

- `default`
 - `youngest`
 - `oldest`
 - `random`
-

election_backoff_ratio

`election_backoff_ratio` specifies the ratio by which master election timeouts increase with each subsequent master election attempt. Defaults to `2`.

election_delay

`election_delay` specifies the seconds a slave replica waits after the master has gracefully exited before holding an election for a new master. A value of 0 or less means an election is not called in this case. Defaults to 30.

election_init_timeout

`election_init_timeout` specifies the initial timeout in seconds when holding an election for a new master. Defaults to 2.

init_rep

`init_rep` specifies whether replication is enabled. To enable replication, set this variable to `true`. Defaults to `false`.

init_txn

`init_txn` specifies whether to use transactions to access this database. Defaults to `false`.

lg_bsize

`lg_bsize` specifies the value of the `lg_bsize` parameter passed to the `set_lg_bsize()` Berkeley DB function. There is no default value.

For more details, see the Berkeley DB documentation, available from <http://www.sleepycat.com/>.

lg_max

`lg_max` specifies the value of the `lg_max` parameter passed to the `set_lg_max()` Berkeley DB function. There is no default value.

For more details, see the Berkeley DB documentation, available from <http://www.sleepycat.com/>.

lk_max_lockers

`lk_max_lockers` specifies the value of the `lk_max_lockers` parameter passed to the `lk_max_lockers()` Berkeley DB function. There is no default value.

For more details, see the Berkeley DB documentation, available from <http://www.sleepycat.com/>.

lk_max_locks

`lk_max_locks` specifies the value of the `lk_max_locks` parameter passed to the `lk_max_locks()` Berkeley DB function. There is no default value.

For more details, see the Berkeley DB documentation, available from <http://www.sleepycat.com/>.

lk_max_objects

`lk_max_objects` specifies the value of the `lk_max_objects` parameter passed to the `lk_max_objects()` Berkeley DB function. There is no default value.

For more details, see the Berkeley DB documentation, available from <http://www.sleepycat.com/>.

log_dir

`log_dir` specifies the directory where the log files are stored; relative paths are relative to `db_home`. The directory must be on a local file system. For maximum performance and reliability, place data files and log files on separate disks, managed by different disk controllers. Defaults to `logs`.

log_stats

`log_stats` specifies whether to log database statistics to the event log during shutdown. Defaults to `false`.

old_log_dir

`old_log_dir` is used in TX mode only, and specifies the directory where the old logs are moved, when `checkpoint_deletes_old_logs` is `false`. Defaults to `old_logs`.

master_heartbeat_interval

`master_heartbeat_interval` specifies the interval in seconds between heartbeats sent by slaves to the master to monitor the health of the master. Setting this variable to 0 disables heartbeat messages. Defaults to 10.

max_buffered_msgs

`max_buffered_msgs` specifies the maximum number of replication messages that can be buffered before being sent. Defaults to 20.

max_buffered_msgs_size

`max_buffered_msgs_size` specifies the maximum size in bytes of replication messages that can be buffered before being sent. Defaults to 10240.

max_elections

`max_elections` specifies the maximum number of attempts to elect a master before giving up. Defaults to 7.

max_log_recs

`max_log_recs` specifies the value of the `max` parameter passed to the `set_rep_request()` Berkeley DB function. There is no default value. For more details, see the Berkeley DB documentation, available from <http://www.sleepycat.com/>.

max_rep_threads

`max_rep_threads` specifies the maximum number of threads used to process replication messages. Defaults to 10.

min_log_recs

`min_log_recs` specifies the value of the `min` parameter passed to the `set_rep_request()` Berkeley DB function. There is no default value. For more details, see the Berkeley DB documentation, available from <http://www.sleepycat.com/>.

mp_mmapsize

`mp_mmapsize` specifies the value of the `mp_mmapsize` parameter passed to the `set_mp_mmapsize()` Berkeley DB function. There is no default value. For more details, see the Berkeley DB documentation, available from <http://www.sleepycat.com/>.

ncache

`ncache` specifies the value of the `ncache` parameter passed to the `set_cachesize()` Berkeley DB function. There is no default value. For more details, see the Berkeley DB documentation, available from <http://www.sleepycat.com/>.

prevent_unilateral_promotion

`prevent_unilateral_promotion` specifies whether a replica can declare itself as a master when there are no other replicas active. Defaults to `false`.

private

`private` specifies whether only one process is permitted to use this environment. Set to `false` when you want to obtain statistics on your database with `db_stat`. Defaults to `true`.

recover_fatal

`recover_fatal` specifies whether to perform a fatal recovery instead of a normal recovery. Defaults to `false`.

rep_limit

`rep_limit` specifies a value in megabyte units used to calculate the values of the `gbytes` and `bytes` parameters passed to the `set_rep_limit()` Berkeley DB function. There is no default value.

For more details, see the Berkeley DB documentation, available from <http://www.sleepycat.com/>.

replica_name

`replica_name` specifies the name of the replica in the replica group. Setting this to an empty string means the ORB name is used as the replica name. Defaults to "".

replica_priority

`replica_priority` specifies the replica's priority during elections for a new master. During an election the most up-to-date replica is elected as the new master.

If there is a tie, the replica priority is used to determine which slave is promoted with higher values taking precedence. If multiple replicas have the same priority, a random selection is made. A priority of 0 means the replica is never promoted. Defaults to 1.

run_deadlock_detector

`run_deadlock_detector` is used in TX mode only, and specifies whether the deadlock detector checks if there is a deadlock, each time a lock conflict occurs. Defaults to `true`.

tmp_dir

`tmp_dir` specifies the directory for temporary files. The directory must be on a local file system. Defaults to `tmp`.

tx_max

`tx_max` is used in TX mode only, and specifies the maximum number of concurrent transactions. Defaults to 20.

verb_all

`verb_all` specifies whether to send verbose diagnostics about any event to the event log. Defaults to `false`.

verb_checkpoint

`verb_checkpoint` specifies whether verbose diagnostics about checkpointing are sent to the event log. Defaults to `false`.

verb_deadlock

`verb_deadlock` specifies whether to send verbose diagnostics about deadlock detection to the event log. Defaults to `false`.

verb_recovery

`verb_recovery` specifies whether to send verbose diagnostics about recovery to the event log. Defaults to `false`.

verb_replication

`verb_replication` specifies whether to send verbose diagnostics about replication to the event log. Defaults to `false`.

verb_waitsfor

`verb_waitsfor` specifies whether to send verbose diagnostics about lock waits to the event log. Defaults to `false`.

plugins:pss_db:envs:*env-name*:dbs:*storage-home-type-id*

Variables in `plugins:pss_db:envs:env-name:dbs:storage-home-type-id` act on the specified storage home—for example, `BankDemoStore/Bank:1.0`.

The following variables are contained in this namespace:

- `file_name`
- `create_file`
- `truncate_file`
- `file_mode`
- `btree`
- `rduonly`
- `bt_minkey`
- `cache_size_bytes`
- `cache_size_gbytes`
- `h_factor`
- `h_nelem`
- `page_size`

file_name

`file_name` specifies a database file that can be shared by several storage home families.

If not specified, the storage home family is stored in its own database file. The name of this file is `storage-home-type-id`, with the following characters replaced with an underscore (`_`): forward slash and backslash (`/` \), colon (`:`), and period (`.`). If specified, the string value must not contain any of the same characters.

create_file

`create_file` specifies whether to create the file for this storage home family, if it does not already exist. Defaults to `true`.

truncate_file

`truncate_file` specifies whether to truncate this storage home family's file. Defaults to `false`.

file_mode

`file_mode` specifies the file mode on UNIX platforms. Defaults to `0`.

btree

`btree` specifies whether a binary tree or a hash map is used. Defaults to `true`.

rdonly

`rdonly` specifies whether this storage home is family read-only. Defaults to `false`.

bt_minkey

`bt_minkey` specifies the minimum number of keys per binary tree page.

cache_size_bytes

`cache_size_bytes` specifies the database cache size in bytes. Defaults to `0`.

cache_size_gbytes

`cache_size_gbytes` specifies the database cache size in gigabytes. Defaults to 0.

h_factor

`h_factor` specifies the hash table density.

h_nelem

`h_nelem` specifies the maximum number of elements in the hash table.

page_size

`page_size` specifies the database page size. Defaults to 0.

plugins:shmiop

The variables in this namespace configure the behavior of the shared memory plugin. It contains the following variables:

- `incoming_connections:hard_limit`
- `incoming_connections:soft_limit`
- `outgoing_connections:hard_limit`
- `outgoing_connections:soft_limit`

incoming_connections:hard_limit

`incoming_connections:hard_limit` specifies the maximum number of incoming (server-side) connections permitted to SHMIOP. SHMIOP does not accept new connections above this limit. Defaults to `-1` (disabled).

incoming_connections:soft_limit

`incoming_connections:soft_limit` specifies the number of connections at which SHMIOP begins closing incoming (server-side) connections. Defaults to `-1` (disabled).

outgoing_connections:hard_limit

`outgoing_connections:hard_limit` specifies the maximum number of outgoing (client-side) connections permitted to the SHMIOP. SHMIOP does not allow new outgoing connections above this limit. Defaults to `-1` (disabled).

outgoing_connections:soft_limit

`outgoing_connections:soft_limit` specifies the number of connections at which SHMIOP begins closing outgoing (client-side) connections. Defaults to `-1` (disabled).

plugins:tlog

The variables in this namespace configure the behavior of the telecom log service. It contains the following variables:

- `direct_persistence`
- `flush_interval`
- `iiop:port`
- `iterator_timeout`
- `max_records`
- `trace:database`
- `trace:events`
- `trace:flush`
- `trace:lifecycle`
- `trace:locks`
- `trace:repository`
- `trace:transactions`

direct_persistence

`direct_persistence` specifies if the service runs using direct or indirect persistence. the default value is `FALSE`, meaning indirect persistence. This should be set to the same value as the collocated notification service.

flush_interval

`flush_interval` specifies the time interval between automated invocations of the flush operation in seconds. Defaults to 300.

iiop:port

`iiop:port` specifies the port that the service listens on when using direct persistence.

iterator_timeout

`iterator_timeout` specifies the maximum lifetime of inactive iterator objects, in seconds. Iterator objects which are inactive longer than the specified time are automatically reaped. The default is zero, which means that inactive iterator objects are never reaped.

max_records

`max_record` specifies the maximum number of records that a `query()` or `retrieve()` operation can return without using an iterator object. Defaults to 100.

trace:database

`trace:database` specifies the amount of information recorded about the behavior of the service's persistent database. Set this value to 1 or greater to enable tracing. The default is 0 which means no information is recorded.

trace:events

`trace:events` specifies the amount of trace information recorded about log generated events. The default is 0.

trace:flush

`trace:flush` specifies the amount of trace information recorded about log flushing. The default is 0.

trace:lifecycle

`trace:lifecycle` specifies the amount of trace information recorded about lifecycle events in the telecom log service such as log object creation and deletion. The default is 0 which means no information is recorded.

trace:locks

`trace:locks` specifies the amount of information recorded about locks on the service's persistent database. The default is 0 .

trace:repository

`trace:repository` specifies the amount of trace information recorded about transactions with the log repository. The default is 0 .

trace:transactions

`trace:transactions` specifies the amount of information recorded about transactions with the service's persistent database. The default is 0 .

plugins:tlog:database

The variables in this namespace control the behavior of the telecom log service's persistent database. This namespace contains the following variables:

- `checkpoint_archive_old_files`
- `checkpoint_deletes_old_logs`
- `checkpoint_interval`
- `checkpoint_min_size`
- `data_dir`
- `db_home`
- `log_dir`
- `lk_max`
- `max_retries`
- `max_sleep_time`
- `tx_max`
- `mode`
- `old_log_dir`
- `private`
- `recover_fatal`
- `sync_transactions`
- `tmp_dir`

checkpoint_archive_old_files

`checkpoint_archive_old_log_files` specifies whether the telecom log service retains archives of the old logs after each checkpoint. When this property is set to `true`, old logs are moved to `old_log_dir`. Defaults to `false`.

checkpoint_deletes_old_logs

`checkpoint_delete_old_logs` specifies whether the telecom log service deletes old log files for its database after each checkpoint. Defaults to `true`.

checkpoint_interval

`checkpoint_interval` specifies, in seconds, the checkpoint interval for posting data from the transaction log file to the telecom log service's database. To disable checkpointing, set this variable to 0. The default is 300.

checkpoint_min_size

`checkpoint_min_size` specifies the minimum amount of data, in kilobytes, to checkpoint at a time. The default is 65536.

data_dir

`data_dir` specifies the directory where the data files are stored; relative paths are relative to `db_home`. The directory must be on a local file system. Defaults to `data`.

db_home

`db_home` specifies the home directory of the Berkeley DB database.

log_dir

`log_dir` specifies the directory where the log files are stored; relative paths are relative to `db_home`. The directory must be on a local file system. For maximum performance and reliability, place data files and log files on separate disks, managed by different disk controllers. Defaults to `logs`.

lk_max

`lk_max` sets the maximum number of locks allowed on the database at one time. The default is 16384.

max_retries

`max_retries` specifies the maximum number of times to retry database transactions before aborting. The default is 0 (infinite).

max_sleep_time

`max_sleep_time` specifies the maximum number of seconds to sleep while waiting for a database transaction to complete. The time between successive retries grows exponentially until this value is reached, that is 1, 2, 4, 8, ... `max_sleep_time`. The default is 256.

tx_max

`tx_max` specifies the maximum number of concurrent database transactions allowed at any one time. This property should be set proportional to the number of persistent proxies. If the number of persistent proxies out paces the number of transactions allowed, performance will degrade. The default is 0 (infinite).

mode

`mode` specifies the file mode on UNIX platforms. Defaults to 0.

old_log_dir

`old_log_dir` specifies the directory into which old transaction log files for the telecom log service's database are moved if `checkpoint_deletes_old_logs` is set to `false`. Defaults to `old_logs`.

private

`private` specifies whether only one process is permitted to use this environment. Set to `false` when you want to obtain statistics on your database with `db_stat`. Defaults to `true`.

recover_fatal

`recover_fatal` determines whether to perform fatal recovery instead of normal recovery. Defaults to `false`.

sync_transactions

`sync_transactions` specifies whether the telecom log service uses synchronous or asynchronous database transactions.

You can set this variable to `true` or `false`:

- `true` (default) specifies using synchronous database transactions. The channel blocks until the transaction is complete.
 - `false` specifies using asynchronous database transactions. The channel issues the transaction and continues.
-

tmp_dir

`tmp_dir` specifies the directory for temporary files. The directory must be on a local file system. Defaults to `tmp`.

plugins:ziop

The variables in this namespace control the behavior of the Orbix ZIOP compression plug-in. ZIOP stands for Zipped Inter-ORB Protocol, which is an proprietary IONA feature. The `ziop` plug-in provides optional compression/decompression of GIOP messages on the wire. This namespace contains the following variables:

- `Classname`
- `shlib_name`

Classname

`ClassName` specifies the Java class in which the Orbix `ziop` compression plugin resides. This is specified as follows:

```
plugins:ziop:ClassName = "com.iona.corba.ziop.ZIOPPlugIn";
```

shlib_name

`shlib_name` specifies the C++ class in which the Orbix `ziop` compression plugin resides. This is specified as follows:

```
plugins:ziop:shlib_name = "it_ziop";
```

For more information on Orbix ZIOP Compression, see ["policies:ziop" on page 178](#).

CORBA Policies

The policies namespace contains configuration variables for CORBA standard policies and IONA-specific policies.

In this chapter

The following topics are discussed in this chapter:

Core Policies	page 154
CORBA Timeout Policies	page 156
Orbix-Specific Timeout Policies	page 157
policies:ajp	page 158
policies:binding_establishment	page 159
policies:egmiop	page 161
policies:giop	page 162
policies:giop:interop_policy	page 164
policies:http(s)	page 167
policies:iiop	page 169
policies:invocation_retry	page 174
policies:shmiop	page 176
policies:well_known_addressing_policy	page 177
policies:ziop	page 178

Core Policies

Configuration variables for core Orbix policies include:

- `non_tx_target_policy`
 - `rebind_policy`
 - `routing_policy_max`
 - `routing_policy_min`
 - `sync_scope_policy`
 - `work_queue_policy`
-

`non_tx_target_policy`

`non_tx_target_policy` specifies the default `NonTxTargetPolicy` value for use when a non-transactional object is invoked within a transaction. Set to one of the following values:

<code>permit</code>	Maps to the <code>NonTxTargetPolicy</code> value <code>PERMIT</code> .
<code>prevent</code>	Maps to the <code>NonTxTargetPolicy</code> value <code>PREVENT</code> .(default)

`rebind_policy`

`rebind_policy` specifies the default value for `RebindPolicy`. Can be one of the following:

`TRANSPARENT`(default)
`NO_REBIND`
`NO_RECONNECT`

`routing_policy_max`

`routing_policy_max` specifies the default maximum value for `RoutingPolicy`. You can set this to one of the following:

`ROUTE_NONE`(default)
`ROUTE_FORWARD`
`ROUTE_STORE_AND_FORWARD`

routing_policy_min

`routing_policy_min` specifies the default minimum value for `RoutingPolicy`. You can set this to one of the following:

```
ROUTE_NONE(default)
ROUTE_FORWARD
ROUTE_STORE_AND_FORWARD
```

sync_scope_policy

`sync_scope_policy` specifies the default value for `SyncScopePolicy`. You can set this to one of the following:

```
SYNC_NONE
SYNC_WITH_TRANSPORT(default)
SYNC_WITH_SERVER
SYNC_WITH_TARGET
```

work_queue_policy

`work_queue_policy` specifies the default `WorkQueue` to use for dispatching `GIOP Requests` and `LocateRequests` when the `WorkQueuePolicy` is not effective. You can set this variable to a string that is resolved using `ORB.resolve_initial_references()`.

For example, to dispatch requests on the internal multi-threaded work queue, this variable should be set to `IT_MultipleThreadWorkQueue`. Defaults to `IT_DirectDispatchWorkQueue`. For more information about `WorkQueue` policies, see the *CORBA Programmer's Guide*.

CORBA Timeout Policies

Orbix supports standard CORBA timeout policies, to enable clients to abort invocations. Orbix also provides proprietary policies, which enable more fine-grained control. Configuration variables for standard CORBA timeout policies include:

- `relative_request_timeout`
- `relative_roundtrip_timeout`

relative_request_timeout

`relative_request_timeout` specifies how much time, in milliseconds, is allowed to deliver a request. Request delivery is considered complete when the last fragment of the GIOP request is sent over the wire to the target object. There is no default value.

The timeout period includes any delay in establishing a binding. This policy type is useful to a client that only needs to limit request delivery time.

relative_roundtrip_timeout

`relative_roundtrip_timeout` specifies how much time, in milliseconds, is allowed to deliver a request and its reply. There is no default value.

The timeout countdown starts with the request invocation, and includes:

- Marshalling in/inout parameters.
- Any delay in transparently establishing a binding.

If the request times out before the client receives the last fragment of reply data, the request is cancelled using a GIOP `CancelRequest` message and all received reply data is discarded.

For more information about standard CORBA timeout policies, see the *CORBA Programmer's Guide*.

Orbix-Specific Timeout Policies

This section lists configuration variables for the Orbix-specific timeout policies. Orbix specific variables in the `policies` namespace include:

- `relative_binding_exclusive_request_timeout`
- `relative_binding_exclusive_roundtrip_timeout`
- `relative_connection_creation_timeout`

`relative_binding_exclusive_request_timeout`

`relative_binding_exclusive_request_timeout` specifies how much time, in milliseconds, is allowed to deliver a request, exclusive of binding attempts. The countdown begins immediately after a binding is obtained for the invocation. There is no default value.

`relative_binding_exclusive_roundtrip_timeout`

`relative_binding_exclusive_roundtrip_timeout` specifies how much time, in milliseconds, is allowed to deliver a request and receive its reply, exclusive of binding attempts. There is no default value.

`relative_connection_creation_timeout`

`relative_connection_creation_timeout` specifies how much time, in milliseconds, is allowed to resolve each address in an IOR, within each binding iteration. Default is 8 seconds.

An IOR can have several `TAG_INTERNET_IOP` (IIOP transport) profiles, each with one or more addresses, while each address can resolve via DNS to multiple IP addresses. Furthermore, each IOR can specify multiple transports, each with its own set of profiles.

This variable applies to each IP address within an IOR. Each attempt to resolve an IP address is regarded as a separate attempt to create a connection.

policies:ajp

This namespace contains variables used to set AJP related policies. It contains the following variables:

- `buffer_sizes_policy:default_buffer_size`
- `buffer_sizes_policy:max_buffer_size`
- `server_address_mode_policy:port_range`

buffer_sizes_policy:default_buffer_size

`buffer_sizes_policy:default_buffer_size` specifies, in bytes, the initial size of the buffers allocated by AJP. Defaults to 4096. This value must be greater than 80 bytes, and must be evenly divisible by 8.

buffer_sizes_policy:max_buffer_size

`buffer_sizes_policy:max_buffer_size` specifies, in bytes, the maximum buffer size permitted by AJP. Defaults to -1 which indicates unlimited size. If not unlimited, this value must be greater than 80.

server_address_mode_policy:port_range

`server_address_mode_policy:port_range` specifies the range of ports that a server uses when there is no well-known addressing policy specified for the port. Specified values take the format of "*from_port*:*to_port*" (for example, "4003:4008").

policies:binding_establishment

Binding establishment is the process of finding a path from a client to the object being invoked. Each binding attempt steps through the bindings listed in the `client_binding_list` configuration variable. The `policies:binding_establishment` namespace contains variables that specify how much effort Orbix puts into establishing a binding. It contains the following variables:

- `backoff_ratio`
- `initial_iteration_delay`
- `max_binding_iterations`
- `max_forwards`
- `relative_expiry`

backoff_ratio

`backoff_ratio` specifies the degree to which delays between binding retries increase from one retry to the next. Defaults to 2.

Between each attempt there is a delay that has a `initial_iteration_delay` of 100 ms, and this increases by the backoff ratio for each subsequent iteration. For example, with a default `backoff_ratio` of 2, the sequence of delays is 100 ms, 200 ms, and 400 ms.

initial_iteration_delay

`initial_iteration_delay` specifies the amount of time, in milliseconds, between the first and second attempt to establish a binding. Defaults to 100 ms.

max_binding_iterations

`max_binding_iterations` specifies the number of times that a client can try to establish a binding before raising a `TRANSIENT` exception. Defaults to 5. To specify unlimited retries, set to -1.

Note: If location forwarding requires that a new binding be established for a forwarded IOR, only one iteration is allowed to bind the new IOR. If the first binding attempt fails, the client reverts to the previous IOR. This allows a load-balancing forwarding agent to redirect the client to a more responsive server.

max_forwards

`max_forwards` specifies the number of forward attempts that are allowed during binding establishment. Defaults to 20. To specify unlimited forward tries, set to -1.

relative_expiry

`relative_expiry` specifies the amount of time, in milliseconds, allowed to establish a binding. There is no default value.

policies:egmiop

The variables in this namespace set policies used to control the behavior of the MIOP transport. It contains the following variable:

- [client_version_policy](#)
- [server_version_policy](#)

client_version_policy

`client_version_policy` specifies the highest GIOP version used by clients. A client uses the version of GIOP specified by this variable, or the version specified in the IOR profile, whichever is lower. Valid values for this variable are: 1.0, 1.1, and 1.2.

For example, the following file-based configuration entry sets the server GIOP version to 1.1.

```
policies:egmiop:server_version_policy="1.1";
```

The following `itadmin` command set this variable:

```
itadmin variable modify -type string -value "1.1"  
policies:egmiop:server_version_policy
```

server_version_policy

`server_version_policy` specifies the GIOP version published in IIO profiles. This variable takes a value of either 1.1 or 1.2. Orbix servers do not publish IIO 1.0 profiles. The default value is 1.2.

policies:giop

The variables in this namespace set policies that control the behavior of bidirectional GIOP. This feature allows callbacks to be made using a connection opened by the client, instead of requiring the server to open a new connection for the callback. The `policies:giop` namespace includes the following variables:

- `"bidirectional_accept_policy"`.
- `"bidirectional_export_policy"`.
- `"bidirectional_gen3_accept_policy"`.
- `"bidirectional_offer_policy"`.

bidirectional_accept_policy

`bidirectional_accept_policy` specifies the behavior of the accept policy used in bidirectional GIOP. On the server side, the `BiDirPolicy::BiDirAcceptPolicy` for the callback invocation must be set to `ALLOW`. You can set this in configuration as follows:

```
policies:giop:bidirectional_accept_policy="ALLOW";
```

This accepts the client's bidirectional offer, and uses an incoming connection for an outgoing request, as long as the policies effective for the invocation are compatible with the connection.

bidirectional_export_policy

`bidirectional_export_policy` specifies the behavior of the export policy used in bidirectional GIOP. A POA used to activate a client-side callback object must have an effective `BiDirPolicy::BiDirExportPolicy` set to `BiDirPolicy::ALLOW`. You can set this in configuration as follows:

```
policies:giop:bidirectional_export_policy="ALLOW";
```

Alternatively, you can do this programmatically by including this policy in the list passed to `POA::create_POA()`.

bidirectional_gen3_accept_policy

`bidirectional_gen3_accept_policy` specifies whether interoperability with Orbix 3.x is enabled. Set this variable to `ALLOW` to enable interoperability with Orbix 3.x:

```
policies:giop:bidirectional_gen3_accept_policy="ALLOW";
```

This allows an Orbix 6.x server to invoke on an Orbix 3.x callback reference in a bidirectional fashion.

bidirectional_offer_policy

`bidirectional_offer_policy` specifies the behavior of the offer policy used in bidirectional GIOP. A bidirectional offer is triggered for an outgoing connection by setting the effective `BiDirPolicy::BiDirOfferPolicy` to `ALLOW` for an invocation. You can set this in configuration as follows:

```
policies:giop:bidirectional_offer_policy="ALLOW";
```

Further information

For more information on all the steps involved in setting bidirectional GIOP, see the *Orbix Administrator's Guide*.

policies:giop:interop_policy

The `policies:giop:interop_policy` child namespace contains variables used to configure interoperability with previous versions of IONA products. It contains the following variables:

- `allow_value_types_in_1_1`
- `cache_is_a`
- `enable_principal_service_context`
- `ignore_message_not_consumed`
- `negotiate_transmission_codeset`
- `send_locate_request`
- `send_principal`

allow_value_types_in_1_1

`allow_value_types_in_1_1` relaxes GIOP 1.1 compliance to allow `valuetypes` to be passed by Java ORBs using GIOP 1.1. This functionality can be important when interoperating with older ORBs that do not support GIOP 1.2. To relax GIOP 1.1 compliance set this variable to `true`.

cache_is_a

`cache_is_a` enables a Java ORB to cache the results of `is_a` invocations, and eliminates the need to make a remote `is_a` callback. The default value is `false`. This feature is Java only.

When passing a derived type as a base type parameter in an IDL operation, the ORB's server-side proxy calls back to the client to confirm that the derived type inherits from the base. For example, take the following IDL:

```
interface BaseType{
    void pass_object(in BaseType obj);
};
interface DerivedType : BaseType {
};
```

Calling `base_object.pass_object(derived_object)` results in the server-side ORB calling back to the client ORB to check that `DerivedType` "is_a" `BaseType`.

This behavior is CORBA compliant, and is performed transparently using an `is_a` callback from the server-side proxy to the client. However, if the client is using a single-threaded POA, and is already invoking on application code, this may result in deadlock. This configuration setting enables the server-side proxy to cache the results of `is_a` invocations, and eliminates the need for a remote `is_a` callback:

```
policies:giop:interop_policy:cache_is_a = "true";
```

Application code can also prime the `is_a` cache with interface type hierarchy information by narrowing the derived type to the base type in application code before potential deadlock would occur. For example, adding the following line to the server mainline primes the cache for the example IDL interfaces:

```
BaseTypeHelper.narrow(derived_object);
```

Applications that frequently pass objects of derived type as base type parameters can also use the `cache_is_a` configuration setting to improve performance.

To maximize type safety and ensure consistent behavior with previous releases, the default value of this variable is `false`.

enable_principal_service_context

`enable_principal_service_context` specifies whether to permit a principal user identifier to be sent in the service context of CORBA requests. This is used to supply an ORB on the mainframe with a user against which basic authorization can take place.

Typically, on the mid-tier, you may want to set the principal to a user that can be authorized on the mainframe. This can be performed on a per-request basis in a portable interceptor. See the *CORBA Programmer's Guide* for how to write portable interceptors.

To enable principal service contexts, set this variable to `true`:

```
policies:giop:interop_policy:enable_principal_service_context="true";
```

ignore_message_not_consumed

`ignore_message_not_consumed` specifies whether to raise `MARSHAL` exceptions when interoperating with ORBs that set message size incorrectly, or with earlier versions of Orbix if it sends piggyback data. The default value is `false`.

The `MARSHAL` exception is set with one of the following minor codes:

- `REQUEST_MESSAGE_NOT_CONSUMED`
 - `REPLY_MESSAGE_NOT_CONSUMED`
-

negotiate_transmission_codeset

`negotiate_transmission_codeset` specifies whether to enable codeset negotiation for wide characters used by some third-party ORBs, previous versions of Orbix, and OrbixWeb. Defaults to `true`.

If this variable is set to `true`, native and conversion codesets for `char` and `wchar` are advertised in `IOP::TAG_CODE_SETS` tagged components in published IORs. The transmission codesets are negotiated by clients and transmitted using an `IOP::CodeSets` service context.

If the variable is `false`, negotiation does not occur and Orbix uses transmission codesets of UTF-16 and ISO-Latin-1 for `wchar` and `char` types, respectively. Defaults to `true`.

send_locate_request

`send_locate_request` specifies whether GIOP sends `LocateRequest` messages before sending initial `Request` messages. Required for interoperability with Orbix 3.0. Defaults to `true`.

send_principal

`send_principal` specifies whether GIOP sends `Principal` information containing the current user name in GIOP 1.0 and GIOP 1.1 requests. Required for interoperability with Orbix 3.0 and Orbix for OS/390. Defaults to `false`.

policies:http(s)

This namespace contains variables used to set HTTP-related policies. It contains the following variables:

- `buffer_sizes_policy:default_buffer_size`
- `buffer_sizes_policy:max_buffer_size`
- `keep-alive:enabled`
- `server_address_mode_policy:port_range`
- `transfer-encoding:chunked:enabled`
- `transfer-encoding:chunked:reserved_buffer_size`

buffer_sizes_policy:default_buffer_size

`buffer_sizes_policy:default_buffer_size` specifies, in bytes, the initial size of the buffers allocated by HTTP. Defaults to 4096. This value must be greater than 80 bytes, and must be evenly divisible by 8.

buffer_sizes_policy:max_buffer_size

`buffer_sizes_policy:max_buffer_size` specifies, in bytes, the maximum buffer size permitted by HTTP. Defaults to -1 which indicates unlimited size. If not unlimited, this value must be greater than 80.

keep-alive:enabled

`keep-alive:enabled` specifies if the server will use persistent connections in response to an incoming `Connection:keep-alive` header. If set to `true`, the server will honor the connection setting from the client. If set to `false`, the server will always ignore the connection setting from the client. If no

connection setting is sent from the client and this variable is set to `true`, the server will respond with `Connection:close` for HTTP 1.0 requests and `Connection:keep-alive` for HTTP 1.1 requests. Defaults to `false`.

Note: Setting this variable to `true` does not prevent the server from ultimately choosing to ignore the keep-alive setting for other reasons. For example if an explicit per client service limit is reached the server will respond with a `Connection:close` regardless of the variable's setting.

server_address_mode_policy:port_range

`server_address_mode_policy:port_range` specifies the range of ports that a server uses when there is no well-known addressing policy specified for the port.

transfer-encoding:chunked:enabled

`transfer-encoding:chunked:enabled` specifies if chunked transfer encoding is enabled. If set to `true`, HTTP messages will be sent as a series of chunks as specified by the HTTP `Transfer-Encoding` header. Each chunk contains: a chunk size specified in base 16, a `CR/LF`, the chunk body, and a closing `CR/LF`. If set to `false`, all HTTP messages sent from Orbix must contain an explicit `Content-Length` header. Defaults to `true`.

transfer-encoding:chunked:reserved_buffer_size

`transfer-encoding:chunked:reserved_buffer_size` specifies the maximum number of bytes reserved in each chunked buffer which may be used to contain the chunk header. The reserved buffer must be at least 8 bytes. Defaults to 8.

policies:iiop

The `policies:iiop` namespace contains variables used to set IIOp-related policies. It contains the following variables:

- `client_address_mode_policy:local_hostname`
- `client_address_mode_policy:port_range`
- `client_version_policy`
- `buffer_sizes_policy:default_buffer_size`
- `buffer_sizes_policy:max_buffer_size`
- `server_address_mode_policy:local_hostname`
- `server_address_mode_policy:port_range`
- `server_address_mode_policy:publish_hostname`
- `server_version_policy`
- `tcp_options_policy:no_delay`
- `tcp_options_policy:recv_buffer_size`
- `tcp_options_policy:send_buffer_size`

client_address_mode_policy:local_hostname

`client_address_mode_policy:local_hostname` specifies the host name that is used by the client.

This variable enables support for *multi-homed* client hosts. These are client machines with multiple hostnames or IP addresses (for example, those using multiple DNS aliases or multiple network interface cards). The `local_hostname` variable enables you to explicitly specify the host name that the client listens on.

For example, if you have a client machine with two network addresses (207.45.52.34 and 207.45.52.35), you can explicitly set this variable to either address:

```
policies:iiop:client_address_mode_policy:local_hostname =
  "207.45.52.34";
```

By default, the `local_hostname` variable is unspecified, and the client uses the `0.0.0.0` wildcard address. In this case, the network interface card used is determined by the operating system.

client_address_mode_policy:port_range

(C++ only) `client_address_mode_policy:port_range` specifies the range of ports that a client uses when there is no well-known addressing policy specified for the port. Specified values take the format of `from_port:to_port`, for example:

```
policies:iiop:client_address_mode_policy:port_range="4003:4008"
```

client_version_policy

`client_version_policy` specifies the highest GIOP version used by clients. A client uses the version of GIOP specified by this variable, or the version specified in the IOR profile, whichever is lower. Valid values for this variable are: 1.0, 1.1, and 1.2.

For example, the following file-based configuration entry sets the server IIOP version to 1.1.

```
policies:iiop:server_version_policy="1.1";
```

The following `itadmin` command set this variable:

```
itadmin variable modify -type string -value "1.1"
  policies:iiop:server_version_policy
```

buffer_sizes_policy:default_buffer_size

`buffer_sizes_policy:default_buffer_size` specifies, in bytes, the initial size of the buffers allocated by IIOp. Defaults to 16000. This value must be greater than 80 bytes, and must be evenly divisible by 8.

buffer_sizes_policy:max_buffer_size

`buffer_sizes_policy:max_buffer_size` specifies the maximum buffer size permitted by IIOp, in kilobytes. Defaults to -1, which indicates unlimited size. If not unlimited, this value must be greater than 80.

server_address_mode_policy:local_hostname

`server_address_mode_policy:local_hostname` specifies the server host name that is advertised by the locator daemon/configuration repository, and listened on by server-side IIOp.

This variable enables support for *multi-homed* server hosts. These are server machines with multiple hostnames or IP addresses (for example, those using multiple DNS aliases or multiple network interface cards). The `local_hostname` variable enables you to explicitly specify the host name that the server listens on and publishes in its IORs.

For example, if you have a machine with two network addresses (207.45.52.34 and 207.45.52.35), you can explicitly set this variable to either address:

```
policies:iio:server_address_mode_policy:local_hostname =  
  "207.45.52.34";
```

By default, the `local_hostname` variable is unspecified. Servers use the default hostname configured for the machine with the Orbix configuration tool.

server_address_mode_policy:port_range

`server_address_mode_policy:port_range` specifies the range of ports that a server uses when there is no well-known addressing policy specified for the port. Specified values take the format of *from_port:to_port*, for example:

```
policies:iiop:server_address_mode_policy:port_range="4003:4008"
```

server_address_mode_policy:publish_hostname

`server_address_mode_policy:publish_hostname` specifies whether IIOP exports hostnames or IP addresses in published profiles. Defaults to `false` (exports IP addresses, and does not export hostnames). To use hostnames in object references, set this variable to `true`, as in the following file-based configuration entry:

```
policies:iiop:server_address_mode_policy:publish_hostname=true
```

The following `itadmin` command is equivalent:

```
itadmin variable create -type bool -value true  
policies:iiop:server_address_mode_policy:publish_hostname
```

server_version_policy

`server_version_policy` specifies the GIOP version published in IIOP profiles. This variable takes a value of either `1.1` or `1.2`. Orbix servers do not publish IIOP 1.0 profiles. The default value is `1.2`.

tcp_options_policy:no_delay

`tcp_options_policy:no_delay` specifies whether the `TCP_NODELAY` option should be set on connections. Defaults to `false`.

tcp_options_policy:recv_buffer_size

`tcp_options_policy:recv_buffer_size` specifies the size of the TCP receive buffer. This variable can only be set to 0, which corresponds to using the default size defined by the operating system.

tcp_options_policy:send_buffer_size

`tcp_options_policy:send_buffer_size` specifies the size of the TCP send buffer. This variable can only be set to 0, which corresponds to using the default size defined by the operating system.

policies:invocation_retry

The `policies:invocation_retry` namespace contains variables that determine how a CORBA ORB reinvokes or rebinds requests that raise the following exceptions:

- `TRANSIENT` with a completion status of `COMPLETED_NO` (triggers transparent reinvocations).
- `COMM_FAILURE` with a completion status of `COMPLETED_NO` (triggers transparent rebinding).

This namespace contains the following variables:

- `backoff_ratio`
- `initial_retry_delay`
- `max_forwards`
- `max_rebinds`
- `max_retries`

backoff_ratio

`backoff_ratio` specifies the degree to which delays between invocation retries increase from one retry to the next. Defaults to 2.

initial_retry_delay

`initial_retry_delay` specifies the amount of time, in milliseconds, between the first and second retries. Defaults to 100.

Note: The delay between the initial invocation and first retry is always 0.

max_forwards

`max_forwards` specifies the number of forward tries allowed for an invocation. Defaults to 20. To specify unlimited forward tries, set to -1.

max_rebinds

`max_rebinds` specifies the number of transparent rebinds attempted on receipt of a `COMM_FAILURE` exception. Defaults to 5.

Note: This setting is valid only if the effective `RebindPolicy` is `TRANSPARENT`; otherwise, no rebinding occurs. For more information, see [“rebind_policy” on page 154](#).

max_retries

`max_retries` specifies the number of transparent reinvocations attempted on receipt of a `TRANSIENT` exception. Defaults to 5.

For more information about proprietary Orbix timeout policies, see the *CORBA Programmer's Guide*.

policies:shmiop

Variables in the `policies:shmiop` namespace set policies related to the shared memory transport (SHMIOP). The following variables are in this namespace:

- `client_version_policy`
- `server_version_policy`

client_version_policy

`client_version_policy` specifies the maximum SHMIOP version used to send IIOP requests. This variable takes a value of either 1.1 or 1.2. Defaults to 1.2.

server_version_policy

`server_version_policy` specifies the SHMIOP version published in SHMIOP profiles. This variable takes a value of either 1.1 or 1.2. Defaults to 1.2.

policies:well_known_addressing_policy

This section describes the configuration variables that specify well-known addressing. These include:

- [http:addr_list](#)
- [https:addr_list](#)
- [ajp13:addr_list](#)

http:addr_list

Provides a list of server names and associated `http` ports. The default value is `[localhost:9000]`.

https:addr_list

Provides a list of server names and associated `https` ports. The default value is `[localhost:9001]`.

ajp13:addr_list

The port number for AJP communication. The default value is `["host-name:6601"]`.

policies:ziop

The variables in this namespace control the behavior of Orbix ZIOP compression. ZIOP stands for Zipped Inter-ORB Protocol, which is an proprietary IONA feature. The `ziop` plug-in provides optional compression/decompression of GIOP messages on the wire. This namespace contains the following variables:

- `compression_enabled`
- `compressor_id`
- `compressor:compressor_id:level`
- `compression_threshold`

compression_enabled

`compression_enabled` specifies whether to enable compression. The default value is `true`:

```
policies:ziop:compression_enabled = "true";
```

This means that even when this entry does not appear in configuration, compression is enabled. However, the `ziop` plug-in must first be loaded in the `orb_plugins` list, and selected by a server or client binding.

compressor_id

`compressor_id` specifies the default compression algorithm. For example:

```
policies:ziop:compressor_id = "1";
```

Possible values are as follows:

- 1 gzip algorithm
- 2 pkzip algorithm
- 3 bzip2 algorithm

If the `compressor_id` is not specified, the default value is 1 (gzip compression).

The ZIOP compression plug-in can be extended with additional compression algorithms using the `IT_ZIOP::CompressionManager` API. See the *Orbix CORBA Programmer's Guide* for details.

compressor:compressor_id:level

`policies:ziop:compressor:compressor_id:level` sets the compression levels. Using this variable, you can specify the compression level for each of the algorithms registered in the `ziop` plug-in. The permitted values are specific to the selected algorithm. For example:

```
policies:ziop:compressor:1:level = "9";
```

For the `gzip` and `pkzip` algorithms, possible values are in the range between 0 (no compression) and 9 (maximum compression). The default value is 9.

For the `bzip2` algorithm, (`compressor_id = 3`), possible values are in the range between 1 (least compression) and 9 (maximum compression). The default value is 9.

compression_threshold

`policies:ziop:compression_threshold` specifies the minimum message size that is compressed. For example:

```
policies:ziop:compression_threshold = "50";
```

Using this setting, messages smaller than 50 bytes are not compressed. The default setting is 0, which means that all messages are compressed.

If you set this to a negative value, the compression threshold is equal to infinity, which means that messages are never compressed. This can be of use if you want to enable compression in one direction only. For example, you can compress messages sent from the server to the client, while in the other direction, messages from the client to the server remain uncompressed.

JMS

The configuration information for IONA's JMS implementation is broken down into several namespaces.

In this chapter

The following topics are discussed in this chapter:

destinations	page 182
factory	page 183
instrumentation	page 184
jmx:adaptor	page 185
persistence	page 186
plugins:jms	page 188

destinations

The variables in this namespace control the destinations that JMS creates on start-up. It contains the following variables:

- `topic_list`
 - `queue_list`
-

topic_list

`topic_list` specifies the names of the initial topic objects JMS creates to support publish and subscribe messages when it starts. Defaults to `["topic0", "topic1"]`.

queue_list

`queue_list` specifies the names of the initial queue objects JMS creates to support point to point messages when it starts. Defaults to `["queue0", "queue1"]`.

factory

The two variables in this namespace allow you to configure a username and password for accessing the JMS `ConnectionFactory` object.

user

`user` specifies the username.

password

`password` specifies the password.

instrumentation

The variables in this namespace control the amount of detail reported to the management service by JMS. It contains the following variables:

- `enabled`
-

enabled

`enabled` specifies if verbose reporting of statistics is activated for the service. Defaults to `false`, which means verbose reporting is disabled.

jmx:adaptor

The variables in this namespace control the reference implementation JMX Web adaptor for JMS. This adaptor is a light-weight alternative to using the management service and is only suitable for testing purposes. The Web adaptor allows monitoring of the JMS management features, using a web browser. It contains the following variables:

- `enabled`
- `port`

enabled

`enabled` specifies if the web adaptor is enabled. Defaults to `false`, which means the web adaptor is disabled.

port

`port` specifies the port number to access the web adaptor. The URL for monitoring JMS is `http://localhost:<port>`.

persistence

The variables in this namespace configure the JMS persistent store. It contains the following variables:

- `message_store`
- `jdbc:driver`
- `jdbc:url`
- `jdbc:user`
- `jdbc:password`
- `jdbc:connection_pool:min`
- `jdbc:connection_pool:max`
- `jdbc:max_message_size`

message_store

`message_store` specifies the name of the database implementation being used as the JMS persistent store. Defaults to "Cloudscape".

jdbc:driver

`jdbc:driver` specifies the driver used to control the persistent store. Defaults to "COM.cloudscape.core.JDBCdriver".

jdbc:url

`jdbc:url` specifies the URL for contacting the persistent store. Defaults to "jdbc:cloudscape:jms;create=true".

jdbc:user

`jdbc:user` specifies the user name to use when accessing the persistent store. Defaults to "".

jdbc:password

`jdbc:password` specifies the password used when accessing the persistent store. Defaults to "".

jdbc:connection_pool:min

`jdbc:connection_pool:min` specifies the minimum number of connection objects available for JMS messages. Defaults to 20.

jdbc:connection_pool:max

`jdbc:connection_pool:max` specifies the maximum number of connection available for JMS messages. Defaults to 20.

jdbc:max_message_size

`jdbc:max_message_size` specifies the upper limit for the size of a JMS message, in bytes.

plugins:jms

The variables in this namespace control the runtime behavior of the JMS broker.

The following variables are contained in this namespace:

- `direct_persistence`
- `iiop:port`
- `is_managed`

direct_persistence

`direct_persistence` specifies if the service runs using direct or indirect persistence. If you deploy JMS into a domain with a locator daemon, the default value is `false`, meaning indirect persistence. It is `true` otherwise.

iiop:port

`iiop:port` specifies the port on which JMS listens on when running in direct persistence mode.

is_managed

`is_managed` specifies if JMS can be managed using the management service. Defaults to `false`, which means the management service cannot manage JMS.

Security

This appendix describes variables used by the IONA Security Framework. The Orbix security infrastructure is highly configurable.

In this appendix

This appendix discusses the following topics:

Applying Constraints to Certificates	page 191
initial_references	page 193
plugins:atli2_tls	page 194
plugins:csi	page 195
plugins:csi	page 195
plugins:gsp	page 196
plugins:https	page 201
plugins:iiop_tls	page 202
plugins:kdm	page 208
plugins:kdm_adm	page 210
plugins:locator	page 211
plugins:schannel	page 212
plugins:security	page 213

policies	page 214
policies:csi	page 220
policies:https	page 223
policies:iioptls	page 228
principal_sponsor	page 238
principal_sponsor:csi	page 242

Applying Constraints to Certificates

Certificate constraints policy

You can use the `CertConstraintsPolicy` to apply constraints to peer X.509 certificates by the default `CertificateValidatorPolicy`. These conditions are applied to the owner's distinguished name (DN) on the first certificate (peer certificate) of the received certificate chain. Distinguished names are made up of a number of distinct fields, the most common being Organization Unit (OU) and Common Name (CN).

Configuration variable

You can specify a list of constraints to be used by `CertConstraintsPolicy` through the `policies:iiop_tls:certificate_constraints_policy` or `policies:https:certificate_constraints_policy` configuration variables. For example:

```
policies:iiop_tls:certificate_constraints_policy =
  [ "CN=Johnny*",OU=[unit1|IT_SSL],O=IONA,C=Ireland,ST=Dublin,L=Earth",
    "CN=Paul*",OU=SSLTEAM,O=IONA,C=Ireland,ST=Dublin,L=Earth",
    "CN=TheOmnipotentOne" ];
```

Constraint language

These are the special characters and their meanings in the constraint list:

*	Matches any text. For example: an* matches ant and anger, but not aunt
[]	Grouping symbols.
	Choice symbol. For example: OU=[unit1 IT_SSL] signifies that if the OU is unit1 or IT_SSL, the certificate is acceptable.
=, !=	Signify equality and inequality respectively.

Example

This is an example list of constraints:

```
policies:iiop_tls:certificate_constraints_policy = [
  "OU=[unit1|IT_SSL],CN=Steve*,L=Dublin",
  "OU=IT_ART*,OU!=IT_ARTtesters,CN=[Jan|Donal],ST=
  Boston" ];
```

This constraint list specifies that a certificate is deemed acceptable if and only if it satisfies one or more of the constraint patterns:

```

If
    The OU is unit1 or IT_SSL
    And
    The CN begins with the text Steve
    And
    The location is Dublin
Then the certificate is acceptable
Else (moving on to the second constraint)
If
    The OU begins with the text IT_ART but isn't IT_ARTtesters
    And
    The common name is either Donal or Jan
    And
    The State is Boston
Then the certificate is acceptable
Otherwise the certificate is unacceptable.

```

The language is like a boolean OR, trying the constraints defined in each line until the certificate satisfies one of the constraints. Only if the certificate fails all constraints is the certificate deemed invalid.

Note that this setting can be sensitive about white space used within it. For example, "CN =" might not be recognized, where "CN=" is recognized.

Distinguished names

For more information on distinguished names, see the *Security Guide*.

initial_references

The `initial_references` namespace contains the following configuration variables:

- [IT_TLS_Toolkit:plugin](#)

IT_TLS_Toolkit:plugin

(Windows only.) This configuration variable enables you to specify the underlying SSL/TLS toolkit to be used by Orbix. It is used in conjunction with the `plugins:baltimore_toolkit:shlib_name` and `plugins:schannel_toolkit:shlib_name` configuration variables to implement SSL/TLS toolkit replaceability.

The default is the Baltimore toolkit.

For example, to specify that an application should use the Schannel SSL/TLS toolkit, you would set configuration variables as follows:

```
initial_references:IT_TLS_Toolkit:plugin = "schannel_toolkit";
plugins:schannel_toolkit:shlib_name = "it_tls_schannel";
```

plugins:atli2_tls

The `plugins:atli2_tls` namespace contains the following variable:

- `use_jsse_tk`

use_jsse_tk

(Java only) Specifies whether or not to use the JSSE/JCE architecture with Orbix Java applications. If `true`, Orbix uses the JSSE/JCE architecture to implement SSL/TLS security; if `false`, Orbix uses the Baltimore SSL/TLS toolkit.

The default is `false`.

plugins:csi

The `policies:csi` namespace includes variables that specify settings for Common Secure Interoperability version 2 (CSIv2):

- `ClassName`
- `shlib_name`

ClassName

`ClassName` specifies the Java class that implements the `csi` plugin. The default setting is:

```
plugins:csi:ClassName = "com.ionacorba.security.csi.CSIPlugin";
```

This configuration setting makes it possible for the Orbix core to load the plugin on demand. Internally, the Orbix core uses a Java class loader to load and instantiate the `csi` class. Plugin loading can be initiated either by including the `csi` in the `orb_plugins` list, or by associating the plugin with an initial reference.

shlib_name

`shlib_name` identifies the shared library (or DLL in Windows) containing the `csi` plugin implementation.

```
plugins:csi:shlib_name = "it_csi_prot";
```

The `csi` plug-in becomes associated with the `it_csi_prot` shared library, where `it_csi_prot` is the base name of the library. The library base name, `it_csi_prot`, is expanded in a platform-dependent manner to obtain the full name of the library file.

plugins:gsp

The `plugins:gsp` namespace includes variables that specify settings for the Generic Security Plugin (GSP). This provides authorization by checking a user's roles against the permissions stored in an action-role mapping file. It includes the following:

- `accept_asserted_authorization_info`
- `action_role_mapping_file`
- `assert_authorization_info`
- `authentication_cache_size`
- `authentication_cache_timeout`
- `authorization_policy_enforcement_point`
- `authorization_policy_store_type`
- `authorization_realm`
- `ClassName`
- `enable_authorization`
- `enable_gssup_sso`
- `enable_user_id_logging`
- `enable_x509_sso`
- `enforce_secure_comms_to_sso_server`
- `enable_security_service_cert_authentication`
- `sso_server_certificate_constraints`
- `use_client_load_balancing`

accept_asserted_authorization_info

If `false`, SAML data is not read from incoming connections. Default is `true`.

action_role_mapping_file

Specifies the action-role mapping file URL. For example:

```
plugins:gsp:action_role_mapping_file =  
  "file:///my/action/role/mapping";
```

assert_authorization_info

If `false`, SAML data is not sent on outgoing connections. Default is `true`.

authentication_cache_size

The maximum number of credentials stored in the authentication cache. If this size is exceeded the oldest credential in the cache is removed.

A value of -1 (the default) means unlimited size. A value of 0 means disable the cache.

authentication_cache_timeout

The time (in seconds) after which a credential is considered *stale*. Stale credentials are removed from the cache and the server must re-authenticate with the Orbix security service on the next call from that user. The cache timeout should be configured to be smaller than the timeout set in the `is2.properties` file (by default, that setting is `is2.sso.session.timeout=600`).

A value of -1 (the default) means an infinite time-out. A value of 0 means disable the cache.

authorization_policy_enforcement_point

Specifies whether access decisions should be made locally (based on cached ACL data) or delegated to the Orbix security service. This variable is meaningful only when the `authorization_policy_store_type` is set to `centralized`.

This configuration variable can have the following values:

- `local`—after retrieving and caching ACL data from the Orbix security service, the GSP plug-in consults only the local cache when making access decisions.
- `centralized`—this option is currently *not* implemented. If you set this option, the application will throw a `CORBA::NO_IMPLEMENT` system exception.

The default is `local`.

authorization_policy_store_type

Specifies whether ACL data should be stored locally (on the same host as the Orbix application) or centrally (on the same host as the Orbix security servier). This configuration variable can have the following values:

- `local`—retrieves ACL data from the local file specified by the `plugins:gsp:action_role_mapping_file` configuration variable.
- `centralized`—retrieves ACL data from the Orbix security service. The Orbix security service must be configured to support centralized ACLs by editing the relevant properties in its `is2.properties` file.

The default is `local`.

authorization_realm

`authorization_realm` specifies the iSF authorization realm to which a server belongs. The value of this variable determines which of a user's roles are considered when making an access control decision.

For example, consider a user that belongs to the `ejb-developer` and `corba-developer` roles within the `Engineering` realm, and to the `ordinary` role within the `Sales` realm. If you set `plugins:gsp:authorization_realm` to `Sales` for a particular server, only the `ordinary` role is considered when making access control decisions (using the `action-role` mapping file).

ClassName

`ClassName` specifies the Java class that implements the `gsp` plugin. This configuration setting makes it possible for the Orbix core to load the plugin on demand. Internally, the Orbix core uses a Java class loader to load and instantiate the `gsp` class. Plugin loading can be initiated either by including the `csi` in the `orb_plugins` list, or by associating the plugin with an initial reference.

enable_authorization

A boolean GSP policy that, when `true`, enables authorization using action-role mapping ACLs in server.

Default is `true`.

enable_gssup_sso

Enables SSO with a username and a password (that is, GSSUP) when set to `true`.

enable_user_id_logging

A boolean variable that enables logging of user IDs on the server side. Default is `false`.

Up until the release of Orbix 6.1 SP1, the GSP plug-in would log messages containing user IDs. For example:

```
[junit] Fri, 28 May 2004 12:17:22.0000000 [SLEEPY:3284]
      (IT_CSI:205) I - User alice authenticated successfully.
```

In some cases, however, it might not be appropriate to expose user IDs in the Orbix log. From Orbix 6.2 onward, the default behavior of the GSP plug-in is changed, so that user IDs are *not* logged by default. To restore the pre-Orbix 6.2 behavior and log user IDs, set this variable to `true`.

enable_x509_sso

Enables certificate-based SSO when set to `true`.

enforce_secure_comms_to_sso_server

Enforces a secure SSL/TLS link between a client and the login service when set to `true`. When this setting is true, the value of the SSL/TLS client secure invocation policy does *not* affect the connection between the client and the login service.

Default is `true`.

`enable_security_service_cert_authentication`

A boolean GSP policy that enables X.509 certificate-based authentication on the server side using the Orbix security service.

Default is `false`.

`sso_server_certificate_constraints`

A special certificate constraints policy that applies *only* to the SSL/TLS connection between the client and the SSO login server. For details of the pattern constraint language, see [“Applying Constraints to Certificates” on page 191](#).

`use_client_load_balancing`

A boolean variable that enables load balancing over a cluster of security services. If an application is deployed in a domain that uses security service clustering, the application should be configured to use *client load balancing* (in this context, *client* means a client of the Orbix security service). See also `policies:iiop_tls:load_balancing_mechanism`.

Default is `true`.

plugins:https

The `plugins:https` namespace contains the following variable:

- [ClassName](#)

ClassName

(Java only) This variable specifies the class name of the `https` plug-in implementation. For example:

```
plugins:https:ClassName = "com.ion.corba.https.HTTPSPlugIn";
```

plugins:iiop_tls

The `plugins:iiop_tls` namespace contains the following variables:

- `buffer_pool:recycle_segments`
- `buffer_pool:segment_preallocation`
- `buffer_pools:max_incoming_buffers_in_pool`
- `buffer_pools:max_outgoing_buffers_in_pool`
- `delay_credential_gathering_until_handshake`
- `enable_iiop_1_0_client_support`
- `incoming_connections:hard_limit`
- `incoming_connections:soft_limit`
- `outgoing_connections:hard_limit`
- `outgoing_connections:soft_limit`
- `tcp_listener:reincarnate_attempts`
- `tcp_listener:reincarnation_retry_backoff_ratio`
- `tcp_listener:reincarnation_retry_delay`

In addition, the `plugins:iiop_tls` namespace contains OS/390 specific variables that are used to configure the source of authentication data and certificates for an application.

- `max_chain_length_policy`
- `hfs_keyring_file_stashfile`
- `hfs_keyring_filename`
- `racf_keyring`

buffer_pool:recycle_segments

(Java only) When this variable is set, the `iiop_tls` plug-in reads this variable's value instead of the `plugins:iiop:buffer_pool:recycle_segments` variable's value.

buffer_pool:segment_preallocation

(Java only) When this variable is set, the `iiop_tls` plug-in reads this variable's value instead of the `plugins:iiop:buffer_pool:segment_preallocation` variable's value.

buffer_pools:max_incoming_buffers_in_pool

(C++ only) When this variable is set, the `iiop_tls` plug-in reads this variable's value instead of the `plugins:iiop:buffer_pools:max_incoming_buffers_in_pool` variable's value.

buffer_pools:max_outgoing_buffers_in_pool

(C++ only) When this variable is set, the `iiop_tls` plug-in reads this variable's value instead of the `plugins:iiop:buffer_pools:max_outgoing_buffers_in_pool` variable's value.

delay_credential_gathering_until_handshake

(Windows and Schannel only) This client configuration variable provides an alternative to using the `principal_sponsor` variables to specify an application's own certificate. When this variable is set to `true` and `principal_sponsor:use_principal_sponsor` is set to `false`, the client delays sending its certificate to a server. The client will wait until the server *explicitly* requests the client to send its credentials during the SSL/TLS handshake.

This configuration variable can be used in conjunction with the `plugins:schannel:prompt_with_credential_choice` configuration variable.

enable_iiop_1_0_client_support

This variable enables client-side interoperability of Orbix SSL/TLS applications with legacy IIOp 1.0 SSL/TLS servers, which do not support IIOp 1.1.

The default value is `false`. When set to `true`, Orbix SSL/TLS searches secure target IIOp 1.0 object references for legacy IIOp 1.0 SSL/TLS tagged component data, and attempts to connect on the specified port.

Note: This variable will not be necessary for most users.

incoming_connections:hard_limit

Specifies the maximum number of incoming (server-side) connections permitted to IIOp. IIOp does not accept new connections above this limit. Defaults to -1 (disabled).

When this variable is set, the `iiop_tls` plug-in reads this variable's value instead of the `plugins:iiop:incoming_connections:hard_limit` variable's value.

Please see the chapter on ACM in the *CORBA Programmer's Guide* for further details.

incoming_connections:soft_limit

Specifies the number of connections at which IIOp should begin closing incoming (server-side) connections. Defaults to -1 (disabled).

When this variable is set, the `iiop_tls` plug-in reads this variable's value instead of the `plugins:iiop:incoming_connections:soft_limit` variable's value.

Please see the chapter on ACM in the *CORBA Programmer's Guide* for further details.

outgoing_connections:hard_limit

When this variable is set, the `iiop_tls` plug-in reads this variable's value instead of the `plugins:iiop:outgoing_connections:hard_limit` variable's value.

outgoing_connections:soft_limit

When this variable is set, the `iiop_tls` plug-in reads this variable's value instead of the `plugins:iiop:outgoing_connections:soft_limit` variable's value.

tcp_listener:reincarnate_attempts

(Windows only)

`plugins:iiop_tls:tcp_listener:reincarnate_attempts` specifies the number of times that a Listener recreates its listener socket after receiving a `SocketException`.

Sometimes a network error may occur, which results in a listening socket being closed. On Windows, you can configure the listener to attempt a reincarnation, which enables new connections to be established. This variable only affects Java and C++ applications on Windows. Defaults to 0 (no attempts).

tcp_listener:reincarnation_retry_backoff_ratio

(Windows only)

`plugins:iiop_tls:tcp_listener:reincarnation_retry_delay` specifies a delay between reincarnation attempts. Data type is `long`. Defaults to 0 (no delay).

tcp_listener:reincarnation_retry_delay

(Windows only)

`plugins:iiop_tls:tcp_listener:reincarnation_retry_backoff_ratio` specifies the degree to which delays between retries increase from one retry to the next. Datatype is `long`. Defaults to 1

hfs_keyring_file_password

`hfs_keyring_file_password` specifies the password that accesses the key database specified by `plugins:iiop_tls:hfs_keyring_filename`.

hfs_keyring_file_stashfile

`hfs_keyring_file_stashfile` specifies the name of a stash file containing the password that accesses the key database specified by `plugins:iiop_tls:hfs_keyring_filename`. The stash file stores the password in encrypted form.

Note: Either `hfs_keyring_file_password` or `hfs_keyring_file_stashfile` can be used to specify the password, but not both.

hfs_keyring_filename

`hfs_keyring_filename` specifies the name of a key ring file (database of keys) within a hierarchical file system. For example, to specify the `/keyring/key.kdb` key ring file:

```
plugins:iiop_tls:hfs_keyring_filename = "/keyring/key.kdb";
```

racf_keyring

`racf_keyring` specifies the name of an RACF key ring from which an application retrieves authentication data. For example, to use the RACF key ring named `TESTRING`:

```
plugins:iiop_tls:racf_keyring = "TESTRING";
```

plugins:kdm

The `plugins:kdm` namespace contains the following variables:

- `cert_constraints`
- `iiop_tls:port`
- `checksums_optional`

cert_constraints

Specifies the list of certificate constraints for principals attempting to open a connection to the KDM server plug-in. See [“Applying Constraints to Certificates” on page 191](#) for a description of the certificate constraint syntax.

To protect the sensitive data stored within it, the KDM applies restrictions on which entities are allowed talk to it. A security administrator should choose certificate constraints that restrict access to the following principals:

- The locator service (requires read-only access).
- The `kdm_admin` plug-in, which is normally loaded into the `itadmin` utility (requires read-write access).

All other principals should be blocked from access. For example, you might define certificate constraints similar to the following:

```
plugins:kdm:cert_constraints =  
  ["C=US,ST=Massachusetts,O=ABigBank*,CN=Secure admin*",  
   "C=US,ST=Boston,O=ABigBank*,CN=Orbix2000 Locator Service*"]
```

Your choice of certificate constraints will depend on the naming scheme for your subject names.

iiop_tls:port

Specifies the well known IP port on which the KDM server listens for incoming calls.

checksums_optional

When equal to `false`, the secure information associated with a server must include a checksum; when equal to `true`, the presence of a checksum is optional. Default is `false`.

plugins:kdm_adm

The `plugins:kdm_adm` namespace contains the following variable:

- `cert_constraints`

cert_constraints

Specifies the list of certificate constraints that are applied when the KDM administration plug-in authenticates the KDM server. See [“Applying Constraints to Certificates” on page 191](#) for a description of the certificate constraint syntax.

The KDM administration plug-in requires protection against attack from applications that try to impersonate the KDM server. A security administrator should, therefore, choose certificate constraints that restrict access to trusted KDM servers only. For example, you might define certificate constraints similar to the following:

```
plugins:kdm_adm:cert_constraints =  
  [ "C=US,ST=Massachusetts,O=ABigBank*,CN=IT_KDM*" ] ;
```

Your choice of certificate constraints will depend on the naming scheme for your subject names.

plugins:locator

The plugins:locator namespace contains the following variable:

- `iiop_tls:port`

iiop_tls:port

Specifies the IP port number where the Orbix locator service listens for secure connections.

Note: This is only useful for applications that have a single TLS listener. For applications that have multiple TLS listeners, you need to programmatically specify the well-known addressing policy.

plugins:schannel

The `plugins:schannel` namespace contains the following variable:

- `prompt_with_credential_choice`

prompt_with_credential_choice

(Windows and Schannel only) Setting both this variable and the `plugins:iiop_tls:delay_credential_gathering_until_handshake` variable to `true` on the client side allows the user to choose which credentials to use for the server connection. The choice of credentials offered to the user is based on the trusted CAs sent to the client in an SSL/TLS handshake message.

If `prompt_with_credential_choice` is set to `false`, Orbix chooses the first certificate it finds in the certificate store that meets the applicable constraints.

The certificate prompt can be replaced by implementing an IDL interface and registering it with the ORB.

plugins:security

The `plugins:security` namespace contains the following variable:

- [share_credentials_across_orbs](#)

share_credentials_across_orbs

Enables own security credentials to be shared across ORBs. Normally, when you specify an own SSL/TLS credential (using the principal sponsor or the principal authenticator), the credential is available only to the ORB that created it. By setting the

`plugins:security:share_credentials_across_orbs` variable to `true`, however, the own SSL/TLS credentials created by one ORB are automatically made available to any other ORBs that are configured to share credentials.

See also `principal_sponsor:csi:use_existing_credentials` for details of how to enable sharing of CSI credentials.

Default is `false`.

policies

The `policies` namespace defines the default CORBA policies for an ORB. Many of these policies can also be set programmatically from within an application. SSL/TLS-specific variables in the `policies` namespace include:

- `allow_unauthenticated_clients_policy`
- `certificate_constraints_policy`
- `client_secure_invocation_policy:requires`
- `client_secure_invocation_policy:supports`
- `max_chain_length_policy`
- `mechanism_policy:ciphersuites`
- `mechanism_policy:protocol_version`
- `session_caching_policy`
- `session_caching`
- `target_secure_invocation_policy:requires`
- `target_secure_invocation_policy:supports`
- `trusted_ca_list_policy`

allow_unauthenticated_clients_policy

(Deprecated in favor of `policies:iiop_tls:allow_unauthenticated_clients_policy` and `policies:https:allow_unauthenticated_clients_policy`.)

A generic variable that sets this policy both for `iiop_tls` and `https`. The recommended alternative is to use the variables prefixed by `policies:iiop_tls` and `policies:https` instead, which take precedence over this generic variable.

certificate_constraints_policy

(Deprecated in favor of

`policies:iiop_tls:certificate_constraints_policy` and
`policies:https:certificate_constraints_policy`.)

A generic variable that sets this policy both for `iiop_tls` and `https`. The recommended alternative is to use the variables prefixed by `policies:iiop_tls` and `policies:https` instead, which take precedence over this generic variable.

client_secure_invocation_policy:requires

(Deprecated in favor of

`policies:iiop_tls:client_secure_invocation_policy:requires` and
`policies:https:client_secure_invocation_policy:requires`.)

A generic variable that sets this policy both for `iiop_tls` and `https`. The recommended alternative is to use the variables prefixed by `policies:iiop_tls` and `policies:https` instead, which take precedence over this generic variable.

client_secure_invocation_policy:supports

(Deprecated in favor of

`policies:iiop_tls:client_secure_invocation_policy:supports` and
`policies:https:client_secure_invocation_policy:supports`.)

A generic variable that sets this policy both for `iiop_tls` and `https`. The recommended alternative is to use the variables prefixed by `policies:iiop_tls` and `policies:https` instead, which take precedence over this generic variable.

max_chain_length_policy

(Deprecated in favor of `policies:iiop_tls:max_chain_length_policy` and `policies:https:max_chain_length_policy`.)

`max_chain_length_policy` specifies the maximum certificate chain length that an ORB will accept. The policy can also be set programmatically using the `IT_TLS_API::MaxChainLengthPolicy` CORBA policy. Default is 2.

Note: The `max_chain_length_policy` is not currently supported on the OS/390 platform.

mechanism_policy:ciphersuites

(Deprecated in favor of `policies:iiop_tls:mechanism_policy:ciphersuites` and `policies:https:mechanism_policy:ciphersuites`.)

`mechanism_policy:ciphersuites` specifies a list of cipher suites for the default mechanism policy. One or more of the cipher suites shown in [Table 5](#) can be specified in this list.

Table 5: *Mechanism Policy Cipher Suites*

Null Encryption, Integrity and Authentication Ciphers	Standard Ciphers
RSA_WITH_NULL_MD5	RSA_EXPORT_WITH_RC4_40_MD5
RSA_WITH_NULL_SHA	RSA_WITH_RC4_128_MD5
	RSA_WITH_RC4_128_SHA
	RSA_EXPORT_WITH_DES40_CBC_SHA
	RSA_WITH_DES_CBC_SHA
	RSA_WITH_3DES_EDE_CBC_SHA

mechanism_policy:protocol_version

(Deprecated in favor of

`policies:iiop_tls:mechanism_policy:protocol_version` and
`policies:https:mechanism_policy:protocol_version`.)

`mechanism_policy:protocol_version` specifies the protocol version used by a security capsule (ORB instance). It can be set to `SSL_V3` or `TLS_V1`. For example:

```
policies:mechanism_policy:protocol_version="TLS_V1"
```

session_caching_policy

(Java only) `session_caching_policy` specifies whether a Java ORB caches the session information for secure associations when acting in a client role, a server role, or both. The purpose of session caching is to enable closed connections to be re-established quickly. The following values are supported:

`CACHE_NONE`(default)
`CACHE_CLIENT`
`CACHE_SERVER`
`CACHE_SERVER_AND_CLIENT`

The policy can also be set programmatically using the `IT_TLS_API::SessionCachingPolicy` CORBA policy.

session_caching

(C++ only) `session_caching` specifies whether a C++ ORB caches the session information for secure associations when acting in a client role, a server role, or both. The purpose of session caching is to enable closed connections to be re-established quickly. The following values are supported:

`CACHE_NONE`(default)

`CACHE_CLIENT`

`CACHE_SERVER`

`CACHE_SERVER_AND_CLIENT`

The policy can also be set programmatically using the `IT_TLS_API::SessionCachingPolicy` CORBA policy.

target_secure_invocation_policy:requires

(Deprecated in favor of

`policies:iiop_tls:target_secure_invocation_policy:requires` and `policies:https:target_secure_invocation_policy:requires`.)

`target_secure_invocation_policy:requires` specifies the minimum level of security required by a server. The value of this variable is specified as a list of association options.

Note: In accordance with CORBA security, this policy cannot be downgraded programmatically by the application.

target_secure_invocation_policy:supports

(Deprecated in favor of

`policies:iiop_tls:target_secure_invocation_policy:supports` and `policies:https:target_secure_invocation_policy:supports`.)

`supports` specifies the maximum level of security supported by a server. The value of this variable is specified as a list of association options. This policy can be upgraded programmatically using either the `QOP` or the `EstablishTrust` policies.

trusted_ca_list_policy

(Deprecated in favor of `policies:iiop_tls:trusted_ca_list_policy` and `policies:https:trusted_ca_list_policy`.)

`trusted_ca_list_policy` specifies a list of filenames, each of which contains a concatenated list of CA certificates in PEM format. The aggregate of the CAs in all of the listed files is the set of trusted CAs.

For example, you might specify two files containing CA lists as follows:

```
policies:trusted_ca_list_policy =  
  ["install_dir/asp/version/etc/tls/x509/ca/ca_list1.pem",  
   "install_dir/asp/version/etc/tls/x509/ca/ca_list_extra.pem"];
```

The purpose of having more than one file containing a CA list is for administrative convenience. It enables you to group CAs into different lists and to select a particular set of CAs for a security domain by choosing the appropriate CA lists.

policies:csi

The `policies:csi` namespace includes variables that specify settings for Common Secure Interoperability version 2 (CSIv2):

- `attribute_service:backward_trust:enabled`
- `attribute_service:client_supports`
- `attribute_service:target_supports`
- `auth_over_transport:authentication_service`
- `auth_over_transport:client_supports`
- `auth_over_transport:server_domain_name`
- `auth_over_transport:target_requires`
- `auth_over_transport:target_supports`

attribute_service:backward_trust:enabled

(Obsolete)

attribute_service:client_supports

`attribute_service:client_supports` is a client-side policy that specifies the association options supported by the CSIv2 attribute service (principal propagation). The only association option that can be specified is `IdentityAssertion`. This policy is normally specified in an intermediate server so that it propagates CSIv2 identity tokens to a target server. For example:

```
policies:csi:attribute_service:client_supports =  
  ["IdentityAssertion"];
```

attribute_service:target_supports

`attribute_service:target_supports` is a server-side policy that specifies the association options supported by the CSIV2 attribute service (principal propagation). The only association option that can be specified is `IdentityAssertion`. For example:

```
policies:csi:attribute_service:target_supports =  
  ["IdentityAssertion"];
```

auth_over_transport:authentication_service

(Java CSI plug-in only) The name of a Java class that implements the `IT_CSI::AuthenticateGSSUPCredentials` IDL interface. The authentication service is implemented as a callback object that plugs into the CSIV2 framework on the server side. By replacing this class with a custom implementation, you could potentially implement a new security technology domain for CSIV2.

By default, if no value for this variable is specified, the Java CSI plug-in uses a default authentication object that always returns `false` when the `authenticate()` operation is called.

auth_over_transport:client_supports

`auth_over_transport:client_supports` is a client-side policy that specifies the association options supported by CSIV2 authorization over transport. The only association option that can be specified is `EstablishTrustInClient`. For example:

```
policies:csi:auth_over_transport:client_supports =  
  ["EstablishTrustInClient"];
```

auth_over_transport:server_domain_name

The iSF security domain (CSlv2 authentication domain) to which this server application belongs. The iSF security domains are administered within an overall security technology domain.

The value of the `server_domain_name` variable will be embedded in the IORs generated by the server. A CSlv2 client about to open a connection to this server would check that the domain name in its own CSlv2 credentials matches the domain name embedded in the IOR.

auth_over_transport:target_requires

`auth_over_transport:target_requires` is a server-side policy that specifies the association options required for CSlv2 authorization over transport. The only association option that can be specified is `EstablishTrustInClient`. For example:

```
policies:csi:auth_over_transport:target_requires =  
  ["EstablishTrustInClient"];
```

auth_over_transport:target_supports

`auth_over_transport:target_supports` is a server-side policy that specifies the association options supported by CSlv2 authorization over transport. The only association option that can be specified is `EstablishTrustInClient`. For example:

```
policies:csi:auth_over_transport:target_supports =  
  ["EstablishTrustInClient"];
```

policies:https

The `policies:https` namespace contains variables used to configure the https plugin. It contains the following variables:

- `allow_unauthenticated_clients_policy`
- `certificate_constraints_policy`
- `client_secure_invocation_policy:requires`
- `client_secure_invocation_policy:supports`
- `max_chain_length_policy`
- `mechanism_policy:ciphersuites`
- `mechanism_policy:protocol_version`
- `session_caching_policy`
- `target_secure_invocation_policy:requires`
- `target_secure_invocation_policy:supports`
- `trusted_ca_list_policy`

allow_unauthenticated_clients_policy

A boolean variable that specifies whether a server will allow a client to establish a secure connection without sending a certificate. Default is `false`.

This configuration variable is applicable *only* in the special case where the target secure invocation policy is set to require `NoProtection` (a semi-secure server).

certificate_constraints_policy

A list of constraints applied to peer certificates—see [“Applying Constraints to Certificates” on page 191](#) for the syntax of the pattern constraint language. If a peer certificate fails to match any of the constraints, the certificate validation step will fail.

The policy can also be set programmatically using the `IT_TLS_API::CertConstraintsPolicy` CORBA policy. Default is no constraints.

client_secure_invocation_policy:requires

Specifies the minimum level of security required by a client. The value of this variable is specified as a list of association options—see the *Orbix Security Guide* for details on how to set SSL/TLS association options.

Note: In accordance with CORBA security, this policy cannot be downgraded programmatically by the application.

client_secure_invocation_policy:supports

Specifies the initial maximum level of security supported by a client. The value of this variable is specified as a list of association options—see the *Orbix Security Guide* for details on how to set SSL/TLS association options.

Note: This policy can be upgraded programmatically using either the `QOP` or the `EstablishTrust` policies.

max_chain_length_policy

The maximum certificate chain length that an ORB will accept (see the discussion of certificate chaining in the *Orbix Security Guide*).

The policy can also be set programmatically using the `IT_TLS_API::MaxChainLengthPolicy` CORBA policy. Default is 2.

Note: The `max_chain_length_policy` is not currently supported on the OS/390 platform.

mechanism_policy:ciphersuites

Specifies a list of cipher suites for the default mechanism policy. One or more of the following cipher suites can be specified in this list:

Table 6: *Mechanism Policy Cipher Suites*

Null Encryption, Integrity and Authentication Ciphers	Standard Ciphers
RSA_WITH_NULL_MD5	RSA_EXPORT_WITH_RC4_40_MD5
RSA_WITH_NULL_SHA	RSA_WITH_RC4_128_MD5
	RSA_WITH_RC4_128_SHA
	RSA_EXPORT_WITH_DES40_CBC_SHA
	RSA_WITH_DES_CBC_SHA
	RSA_WITH_3DES_EDE_CBC_SHA

mechanism_policy:protocol_version

Specifies the protocol version used by a security capsule (ORB instance). Can be set to one of the following values:

TLS_V1
SSL_V3
SSL_V2V3

The `SSL_V2V3` value is a special setting that facilitates interoperability with an Orbix application deployed on the OS/390 platform. Orbix security on the OS/390 platform is based on IBM's System/SSL toolkit, which implements SSL version 3, but does so by using SSL version 2 hellos as part of the handshake. This form of handshake causes interoperability problems, because applications on other platforms identify the handshake as an SSL version 2 handshake.

The misidentification of the SSL protocol version can be avoided by setting the protocol version to be `SSL_V2V3` in the non-OS/390 application (this bug also affects some old versions of Microsoft Internet Explorer).

For example:

```
policies:mechanism_policy:protocol_version = "SSL_V2V3";
```

session_caching_policy

When this policy is set, the `https` plug-in reads this policy's value instead of the [policies:session_caching](#) policy's value (C++) or [policies:session_caching_policy](#) policy's value (Java).

target_secure_invocation_policy:requires

Specifies the minimum level of security required by a server. The value of this variable is specified as a list of association options—see the *Orbix Security Guide* for more details about association options.

In accordance with CORBA security, this policy cannot be downgraded programmatically by the application.

target_secure_invocation_policy:supports

Specifies the maximum level of security supported by a server. The value of this variable is specified as a list of association options—see the *Orbix Security Guide* for more details about association options.

This policy can be upgraded programmatically using either the `QOP` or the `EstablishTrust` policies.

trusted_ca_list_policy

Contains a list of filenames (or a single filename), each of which contains a concatenated list of CA certificates in PEM format. The aggregate of the CAs in all of the listed files is the set of trusted CAs.

For example, you might specify two files containing CA lists as follows:

```
policies:trusted_ca_list_policy =  
  ["ASPInstallDir/asp/6.0/etc/tls/x509/ca/ca_list1.pem",  
   "ASPInstallDir/asp/6.0/etc/tls/x509/ca/ca_list_extra.pem"];
```

The purpose of having more than one file containing a CA list is for administrative convenience. It enables you to group CAs into different lists and to select a particular set of CAs for a security domain by choosing the appropriate CA lists.

policies:iiop_tls

The `policies:iiop_tls` namespace contains variables used to set IIOP-related policies for a secure environment. These settings affect the `iiop_tls` plugin. It contains the following variables:

- `allow_unauthenticated_clients_policy`
- `buffer_sizes_policy:default_buffer_size`
- `buffer_sizes_policy:max_buffer_size`
- `certificate_constraints_policy`
- `client_secure_invocation_policy:requires`
- `client_secure_invocation_policy:supports`
- `client_version_policy`
- `connection_attempts`
- `connection_retry_delay`
- `load_balancing_mechanism`
- `max_chain_length_policy`
- `mechanism_policy:ciphersuites`
- `mechanism_policy:protocol_version`
- `server_address_mode_policy:local_domain`
- `server_address_mode_policy:local_hostname`
- `server_address_mode_policy:port_range`
- `server_address_mode_policy:publish_hostname`
- `server_version_policy`
- `session_caching_policy`
- `target_secure_invocation_policy:requires`
- `target_secure_invocation_policy:supports`
- `tcp_options_policy:no_delay`
- `tcp_options_policy:recv_buffer_size`
- `tcp_options_policy:send_buffer_size`
- `trusted_ca_list_policy`

allow_unauthenticated_clients_policy

A boolean variable that specifies whether a server will allow a client to establish a secure connection without sending a certificate. Default is `false`. This configuration variable is applicable *only* in the special case where the target secure invocation policy is set to require `NoProtection` (a semi-secure server).

buffer_sizes_policy:default_buffer_size

When this policy is set, the `iiop_tls` plug-in reads this policy's value instead of the `policies:iiop:buffer_sizes_policy:default_buffer_size` policy's value.

`buffer_sizes_policy:default_buffer_size` specifies, in bytes, the initial size of the buffers allocated by IIOP. Defaults to 16000. This value must be greater than 80 bytes, and must be evenly divisible by 8.

buffer_sizes_policy:max_buffer_size

When this policy is set, the `iiop_tls` plug-in reads this policy's value instead of the `policies:iiop:buffer_sizes_policy:max_buffer_size` policy's value.

`buffer_sizes_policy:max_buffer_size` specifies the maximum buffer size permitted by IIOP, in kilobytes. Defaults to 512. A value of -1 indicates unlimited size. If not unlimited, this value must be greater than 80.

certificate_constraints_policy

A list of constraints applied to peer certificates—see the discussion of certificate constraints in the Orbix security guide for the syntax of the pattern constraint language. If a peer certificate fails to match any of the constraints, the certificate validation step will fail.

The policy can also be set programmatically using the `IT_TLS_API::CertConstraintsPolicy` CORBA policy. Default is no constraints.

client_secure_invocation_policy:requires

Specifies the minimum level of security required by a client. The value of this variable is specified as a list of association options—see the *Orbix Security Guide* for more details about association options.

In accordance with CORBA security, this policy cannot be downgraded programmatically by the application.

client_secure_invocation_policy:supports

Specifies the initial maximum level of security supported by a client. The value of this variable is specified as a list of association options—see the *Orbix Security Guide* for more details about association options.

This policy can be upgraded programmatically using either the `QOP` or the `EstablishTrust` policies.

client_version_policy

`client_version_policy` specifies the highest IOP version used by clients. A client uses the version of IOP specified by this variable, or the version specified in the IOR profile, whichever is lower. Valid values for this variable are: 1.0, 1.1, and 1.2.

For example, the following file-based configuration entry sets the server IOP version to 1.1.

```
policies:iop:server_version_policy="1.1";
```

The following `itadmin` command set this variable:

```
itadmin variable modify -type string -value "1.1"  
policies:iop:server_version_policy
```

connection_attempts

`connection_attempts` specifies the number of connection attempts used when creating a connected socket using a Java application. Defaults to 5.

connection_retry_delay

`connection_retry_delay` specifies the delay, in seconds, between connection attempts when using a Java application. Defaults to 2.

load_balancing_mechanism

Specifies the load balancing mechanism for the client of a security service cluster (see also `plugins:gsp:use_client_load_balancing`). In this context, a client can also be an *Orbix* server. This policy only affects connections made using IORs that contain multiple addresses. The `iiop_tls` plug-in load balances over the addresses embedded in the IOR.

The following mechanisms are supported:

- `random`—choose one of the addresses embedded in the IOR at random (this is the default).
- `sequential`—choose the first address embedded in the IOR, moving on to the next address in the list only if the previous address could not be reached.

max_chain_length_policy

This policy overrides `policies:max_chain_length_policy` for the `iiop_tls` plugin.

The maximum certificate chain length that an ORB will accept.

The policy can also be set programmatically using the `IT_TLS_API::MaxChainLengthPolicy` CORBA policy. Default is 2.

Note: The `max_chain_length_policy` is not currently supported on the OS/390 platform.

mechanism_policy:ciphersuites

This policy overrides `policies:mechanism_policy:ciphersuites` for the `iiop_tls` plugin.

Specifies a list of cipher suites for the default mechanism policy. One or more of the following cipher suites can be specified in this list:

Table 7: *Mechanism Policy Cipher Suites*

Null Encryption, Integrity and Authentication Ciphers	Standard Ciphers
RSA_WITH_NULL_MD5	RSA_EXPORT_WITH_RC4_40_MD5
RSA_WITH_NULL_SHA	RSA_WITH_RC4_128_MD5
	RSA_WITH_RC4_128_SHA
	RSA_EXPORT_WITH_DES40_CBC_SHA
	RSA_WITH_DES_CBC_SHA
	RSA_WITH_3DES_EDE_CBC_SHA

mechanism_policy:protocol_version

This policy overrides `policies:mechanism_policy:protocol_version` for the `iiop_tls` plugin.

Specifies the protocol version used by a security capsule (ORB instance).

Can be set to one of the following values:

TLS_V1
 SSL_V3
 SSL_V2V3

The `SSL_V2V3` value is a special setting that facilitates interoperability with an Orbix application deployed on the OS/390 platform. Orbix security on the OS/390 platform is based on IBM's System/SSL toolkit, which implements SSL version 3, but does so by using SSL version 2 hellos as part of the handshake. This form of handshake causes interoperability problems, because applications on other platforms identify the handshake as an SSL version 2 handshake. The misidentification of the SSL protocol version can be avoided by setting the protocol version to be `SSL_V2V3` in the non-OS/390 application (this bug also affects some old versions of Microsoft Internet Explorer).

For example:

```
policies:mechanism_policy:protocol_version = "SSL_V2V3";
```

server_address_mode_policy:local_domain

(Java only) When this policy is set, the `iiop_tls` plug-in reads this policy's value instead of the

`policies:iiop:server_address_mode_policy:local_domain` policy's value.

server_address_mode_policy:local_hostname

(Java only) When this policy is set, the `iiop_tls` plug-in reads this policy's value instead of the

`policies:iiop:server_address_mode_policy:local_hostname` policy's value.

`server_address_mode_policy:local_hostname` specifies the hostname advertised by the locator daemon/configuration repository, and listened on by server-side IIOP.

Some machines have multiple hostnames or IP addresses (for example, those using multiple DNS aliases or multiple network cards). These machines are often termed *multi-homed hosts*. The `local_hostname` variable supports these type of machines by enabling you to explicitly specify the host that servers listen on and publish in their IORs.

For example, if you have a machine with two network addresses (207.45.52.34 and 207.45.52.35), you can explicitly set this variable to either address:

```
policies:iiop:server_address_mode_policy:local_hostname =  
  "207.45.52.34";
```

By default, the `local_hostname` variable is unspecified. Servers use the default hostname configured for the machine with the Orbix configuration tool.

server_address_mode_policy:port_range

(Java only) When this policy is set, the `iiop_tls` plug-in reads this policy's value instead of the

`policies:iiop:server_address_mode_policy:port_range` policy's value.

`server_address_mode_policy:port_range` specifies the range of ports that a server uses when there is no well-known addressing policy specified for the port.

server_address_mode_policy:publish_hostname

When this policy is set, the `iiop_tls` plug-in reads this policy's value instead of the

`policies:iiop:server_address_mode_policy:publish_hostname` policy's value.

`server_address_mode_policy:publish_hostname` specifies whether IIOp exports hostnames or IP addresses in published profiles. Defaults to `false` (exports IP addresses, and does not export hostnames). To use hostnames in object references, set this variable to `true`, as in the following file-based configuration entry:

```
policies:iiop:server_address_mode_policy:publish_hostname=true
```

The following `itadmin` command is equivalent:

```
itadmin variable create -type bool -value true
policies:iiop:server_address_mode_policy:publish_hostname
```

server_version_policy

When this policy is set, the `iiop_tls` plug-in reads this policy's value instead of the `policies:iiop:server_version_policy` policy's value.

`server_version_policy` specifies the GIOP version published in IIOp profiles. This variable takes a value of either `1.1` or `1.2`. Orbix servers do not publish IIOp 1.0 profiles. The default value is `1.2`.

session_caching_policy

This policy overrides `policies:session_caching_policy`(Java) and `policies:session_caching`(C++) for the `iiop_tls` plugin.

target_secure_invocation_policy:requires

This policy overrides

`policies:target_secure_invocation_policy:requires` for the `iiop_tls` plugin.

Specifies the minimum level of security required by a server. The value of this variable is specified as a list of association options—see the *Orbix Security Guide* for more details about association options.

In accordance with CORBA security, this policy cannot be downgraded programmatically by the application.

target_secure_invocation_policy:supports

This policy overrides

`policies:target_secure_invocation_policy:supports` for the `iiop_tls` plugin.

Specifies the maximum level of security supported by a server. The value of this variable is specified as a list of association options—see the *Orbix Security Guide* for more details about association options.

This policy can be upgraded programmatically using either the `QOP` or the `EstablishTrust` policies.

tcp_options_policy:no_delay

When this policy is set, the `iiop_tls` plug-in reads this policy's value instead of the `policies:iiop:tcp_options_policy:no_delay` policy's value.

`tcp_options_policy:no_delay` specifies whether the `TCP_NODELAY` option should be set on connections. Defaults to `false`.

tcp_options_policy:recv_buffer_size

When this policy is set, the `iiop_tls` plug-in reads this policy's value instead of the `policies:iiop:tcp_options_policy:recv_buffer_size` policy's value.

`tcp_options_policy:recv_buffer_size` specifies the size of the TCP receive buffer. This variable can only be set to 0, which corresponds to using the default size defined by the operating system.

tcp_options_policy:send_buffer_size

When this policy is set, the `iiop_tls` plug-in reads this policy's value instead of the `policies:iiop:tcp_options_policy:send_buffer_size` policy's value.

`tcp_options_policy:send_buffer_size` specifies the size of the TCP send buffer. This variable can only be set to 0, which corresponds to using the default size defined by the operating system.

trusted_ca_list_policy

This policy overrides the `policies:trusted_ca_list_policy` for the `iiop_tls` plugin.

Contains a list of filenames (or a single filename), each of which contains a concatenated list of CA certificates in PEM format. The aggregate of the CAs in all of the listed files is the set of trusted CAs.

For example, you might specify two files containing CA lists as follows:

```
policies:trusted_ca_list_policy =  
  [ "ASPInstallDir/asp/6.0/etc/tls/x509/ca/ca_list1.pem",  
    "ASPInstallDir/asp/6.0/etc/tls/x509/ca/ca_list_extra.pem" ];
```

The purpose of having more than one file containing a CA list is for administrative convenience. It enables you to group CAs into different lists and to select a particular set of CAs for a security domain by choosing the appropriate CA lists.

principal_sponsor

The `principal_sponsor` namespace stores configuration information to be used when obtaining credentials. Orbix provides an implementation of a principal sponsor that creates credentials for applications automatically. The principal sponsor automatically calls the `authenticate()` operation on the `PrincipalAuthenticator` object after determining the data to supply.

Use of the `PrincipalSponsor` is disabled by default and can only be enabled through configuration.

The `PrincipalSponsor` represents an entry point into the secure system. It must be activated and authenticate the user, before any application-specific logic executes. This allows unmodified, security-unaware applications to have `Credentials` established transparently, prior to making invocations.

In this section

The following variables are in this namespace:

- `use_principal_sponsor`
- `auth_method_id`
- `auth_method_data`
- `callback_handler:ClassName`
- `login_attempts`

use_principal_sponsor

`use_principal_sponsor` specifies whether an attempt is made to obtain credentials automatically. Defaults to `false`. If set to `true`, the following `principal_sponsor` variables must contain data in order for anything to actually happen.

auth_method_id

`auth_method_id` specifies the authentication method to be used. The following authentication methods are available:

<code>pkcs12_file</code>	The authentication method uses a PKCS#12 file.
<code>pkcs11</code>	Java only. The authentication data is provided by a smart card.
<code>security_label</code>	Windows and Schannel only. The authentication data is specified by supplying the common name (CN) from an application certificate's subject DN.

For example, you can select the `pkcs12_file` authentication method as follows:

```
principal_sponsor:auth_method_id = "pkcs12_file";
```

auth_method_data

`auth_method_data` is a string array containing information to be interpreted by the authentication method represented by the `auth_method_id`.

For the `pkcs12_file` authentication method, the following authentication data can be provided in `auth_method_data`:

<code>filename</code>	A PKCS#12 file that contains a certificate chain and private key— <i>required</i> .
<code>password</code>	A password for the private key— <i>optional</i> . It is bad practice to supply the password from configuration for deployed systems. If the password is not supplied, the user is prompted for it.
<code>password_file</code>	The name of a file containing the password for the private key— <i>optional</i> . This option is not recommended for deployed systems.

For the `pkcs11` (smart card) authentication method, the following authentication data can be provided in `auth_method_data`:

<code>provider</code>	A name that identifies the underlying PKCS #11 toolkit used by Orbix to communicate with the smart card. The toolkit currently used by Orbix has the provider name <code>dkck132.dll</code> (from Baltimore).
<code>slot</code>	The number of a particular slot on the smart card (for example, 0) containing the user's credentials.
<code>pin</code>	A PIN to gain access to the smart card— <i>optional</i> . It is bad practice to supply the PIN from configuration for deployed systems. If the PIN is not supplied, the user is prompted for it.

For the `security_label` authentication method on Windows, the following authentication data can be provided in `auth_method_data`:

<code>label</code>	(Windows and Schannel only.) The common name (CN) from an application certificate's subject DN
--------------------	--

For example, to configure an application on Windows to use a certificate, `bob.p12`, whose private key is encrypted with the `bobpass` password, set the `auth_method_data` as follows:

```
principal_sponsor:auth_method_data =
  ["filename=c:\users\bob\bob.p12", "password=bobpass"];
```

The following points apply to Java implementations:

- If the file specified by `filename=` is not found, it is searched for on the classpath.
- The file specified by `filename=` can be supplied with a URL instead of an absolute file location.
- The mechanism for prompting for the password if the password is supplied through `password=` can be replaced with a custom mechanism, as demonstrated by the `login` demo.

- There are two extra configuration variables available as part of the `principal_sponsor` namespace, namely `principal_sponsor:callback_handler` and `principal_sponsor:login_attempts`. These are described below.
 - These Java-specific features are available subject to change in future releases; any changes that can arise probably come from customer feedback on this area.
-

callback_handler:ClassName

`callback_handler:ClassName` specifies the class name of an interface that implements the interface `com.ionacorba.tls.auth.CallbackHandler`. This variable is only used for Java clients.

login_attempts

`login_attempts` specifies how many times a user is prompted for authentication data (usually a password). It applies for both internal and custom `CallbackHandlers`; if a `CallbackHandler` is supplied, it is invoked upon up to `login_attempts` times as long as the `PrincipalAuthenticator` returns `SecAuthFailure`. This variable is only used by Java clients.

principal_sponsor:csi

The `principal_sponsor:csi` namespace stores configuration information to be used when obtaining CSI (Common Secure Interoperability) credentials. It includes the following:

- `use_existing_credentials`
 - `use_principal_sponsor`
 - `auth_method_data`
 - `auth_method_id`
-

use_existing_credentials

A boolean value that specifies whether ORBs that share credentials can also share CSI credentials. If `true`, any CSI credentials loaded by one credential-sharing ORB can be used by other credential-sharing ORBs loaded after it; if `false`, CSI credentials are not shared.

This variable has no effect, unless the `plugins:security:share_credentials_across_orbs` variable is also `true`. Default is `false`.

use_principal_sponsor

`use_principal_sponsor` is a boolean value that switches the CSI principal sponsor on or off.

If set to `true`, the CSI principal sponsor is enabled; if `false`, the CSI principal sponsor is disabled and the remaining `principal_sponsor:csi` variables are ignored. Defaults to `false`.

auth_method_data

`auth_method_data` is a string array containing information to be interpreted by the authentication method represented by the `auth_method_id`.

For the GSSUPMech authentication method, the following authentication data can be provided in `auth_method_data`:

<code>username</code>	The username for CSIV2 authorization. This is optional. Authentication of CSIV2 usernames and passwords is performed on the server side. The administration of usernames depends on the particular security mechanism that is plugged into the server side see auth_over_transport:authentication_service .
<code>password</code>	The password associated with username. This is optional. It is bad practice to supply the password from configuration for deployed systems. If the password is not supplied, the user is prompted for it.
<code>domain</code>	The CSIV2 authentication domain in which the username/password pair is authenticated. When the client is about to open a new connection, this domain name is compared with the domain name embedded in the relevant IOR (see policies:csi:auth_over_transport:server_domain_name). The domain names must match. Note: If <code>domain</code> is an empty string, it matches any target domain. That is, an empty domain string is equivalent to a wildcard.

If any of the preceding data are omitted, the user is prompted to enter authentication data when the application starts up.

For example, to log on to a CSIV2 application as the `administrator` user in the `US-SantaClara` domain:

```
principal_sponsor:csi:auth_method_data =  
  ["username=administrator", "domain=US-SantaClara"];
```

When the application is started, the user is prompted for the administrator password.

Note: It is currently not possible to customize the login prompt associated with the CSIv2 principal sponsor. As an alternative, you could implement your own login GUI by programming and pass the user input directly to the principal authenticator.

auth_method_id

`auth_method_id` specifies a string that selects the authentication method to be used by the CSI application. The following authentication method is available:

<code>GSSUPMech</code>	The Generic Security Service Username/Password (GSSUP) mechanism.
------------------------	---

For example, you can select the `GSSUPMech` authentication method as follows:

```
principal_sponsor:csi:auth_method_id = "GSSUPMech";
```

XA Resource Manager

The XA plugin uses configuration variables in the *rm-name* namespace, where *rm-name* is the name of the resource manager passed to `create_resource_manager()` and `connect_to_resource_manager()` from the `IT_XA::Connector` interface. Therefore, configuration variables for the XA plugin take the form *rm-name:variable_name*. For example to specify the POA name to use for recoverable objects in the resource manager `goliath`, set the configuration variable:

```
goliath:poa_name
```

The following variables are in this namespace:

- `supports_async_rollback`
- `ping_period`
- `open_string`
- `close_string`
- `rmid`

`poa_name`

`poa_name` specifies the persistent POA used by the XA plugin for recoverable objects. Defaults to *rm-name*.

supports_async_rollback

`supports_async_rollback` specifies whether the resource manager allows asynchronous rollbacks—that is, calls to `xa_rollback()` when no transaction is associated with the connection. Defaults to `false`.

ping_period

`ping_period` specifies the time, in seconds, between checking that a transaction is still active. Defaults to 0.

open_string

`open_string` specifies the default open string for the resource manager used during calls to `xa_open()`. Defaults to an empty string.

close_string

`close_string` specifies the default close string for the resource manager used during calls to `xa_close()`. Defaults to an empty string.

rmid

`rmid` specifies the resource manager identifier used for this resource manager. If not set, the XA plugin allocates one.

Glossary

A

administration

All aspects of installing, configuring, deploying, monitoring, and managing a system.

ART

Adaptive Runtime Technology. IONA's modular, distributed object architecture, which supports dynamic deployment and configuration of services and application code. ART provides the foundation for IONA software products.

ATLI2

Abstract Transport Layer Interface, version 2. IONA's current transport layer implementation.

C

Certificate Authority

Certificate Authority (CA). A trusted third-party organization or company that issues digital certificates used to create digital signatures and public-private key pairs. The role of the CA in this process is to guarantee that the individual granted the unique certificate is, in fact, who he or she claims to be. CAs are a crucial component in data security and electronic commerce because they guarantee that the two parties exchanging information are really who they claim to be.

CFR

See [configuration repository](#).

client

An application (process) that typically runs on a desktop and requests services from other applications that often run on different machines (known as server processes). In CORBA, a client is a program that requests services from CORBA objects.

configuration

A specific arrangement of system elements and settings.

configuration domain

Contains all the configuration information that Orbix ORBs, services and applications use. Defines a set of common configuration settings that specify available services and control ORB behavior. This information consists of configuration variables and their values. Configuration domain data can be implemented and maintained in a centralized Orbix configuration repository or as a set of files distributed among domain hosts. Configuration domains let you organize ORBs into manageable groups, thereby bringing scalability and ease of use to the largest environments. See also [configuration file](#) and [configuration repository](#).

configuration file

A file that contains configuration information for Orbix components within a specific configuration domain. See also [configuration domain](#).

configuration repository

A centralized store of configuration information for all Orbix components within a specific configuration domain. See also [configuration domain](#).

configuration scope

Orbix configuration is divided into scopes. These are typically organized into a root scope and a hierarchy of nested scopes, the fully-qualified names of which map directly to ORB names. By organizing configuration properties into various scopes, different settings can be provided for individual ORBs, or common settings for groups of ORB. Orbix services, such as the naming service, have their own configuration scopes.

CORBA

Common Object Request Broker Architecture. An open standard that enables objects to communicate with one another regardless of what programming language they are written in, or what operating system they run on. The CORBA specification is produced and maintained by the OMG. See also [OMG](#).

CORBA naming service

An implementation of the OMG Naming Service Specification. Describes how applications can map object references to names. Servers can register object references by name with a naming service repository, and can advertise those

names to clients. Clients, in turn, can resolve the desired objects in the naming service by supplying the appropriate name. The Orbix naming service is an example.

CORBA objects

Self-contained software entities that consist of both data and the procedures to manipulate that data. Can be implemented in any programming language that CORBA supports, such as C++ and Java.

CORBA transaction service

An implementation of the OMG Transaction Service Specification. Provides interfaces to manage the demarcation of transactions and the propagation of transaction contexts. Orbix OTS is such as service.

CSiv2

The OMG's Common Secure Interoperability protocol v2.0, which can be used to provide the basis for application-level security in both CORBA and J2EE applications. The IONA Security Framework implements CSiv2 to transmit usernames and passwords, and to assert identities between applications.

D**deployment**

The process of distributing a configuration or system element into an environment.

H**HTTP**

HyperText Transfer Protocol. The underlying protocol used by the World Wide Web. It defines how files (text, graphic images, video, and other multimedia files) are formatted and transmitted. Also defines what actions Web servers and browsers should take in response to various commands. HTTP runs on top of TCP/IP.

I

IDL

Interface Definition Language. The CORBA standard declarative language that allows a programmer to define interfaces to CORBA objects. An IDL file defines the public API that CORBA objects expose in a server application. Clients use these interfaces to access server objects across a network. IDL interfaces are independent of operating systems and programming languages.

IFR

See [interface repository](#).

IIOIP

Internet Inter-ORB Protocol. The CORBA standard messaging protocol, defined by the OMG, for communications between ORBs and distributed applications. IIOIP is defined as a protocol layer above the transport layer, TCP/IP.

implementation repository

A database of available servers, it dynamically maps persistent objects to their server's actual address. Keeps track of the servers available in a system and the hosts they run on. Also provides a central forwarding point for client requests. See also [location domain](#) and [locator daemon](#).

IMR

See [implementation repository](#).

installation

The placement of software on a computer. Installation does not include configuration unless a default configuration is supplied.

Interface Definition Language

See [IDL](#).

interface repository

Provides centralized persistent storage of IDL interfaces. An Orbix client can query this repository at runtime to determine information about an object's interface, and then use the Dynamic Invocation Interface (DII) to make calls to the object. Enables Orbix clients to call operations on IDL interfaces that are unknown at compile time.

invocation

A request issued on an already active software component.

IOR

Interoperable Object Reference. See [object reference](#).

L**location domain**

A collection of servers under the control of a single locator daemon. Can span any number of hosts across a network, and can be dynamically extended with new hosts. See also [locator daemon](#) and [node daemon](#).

locator daemon

A server host facility that manages an implementation repository and acts as a control center for a location domain. Orbix clients use the locator daemon, often in conjunction with a naming service, to locate the objects they seek. Together with the implementation repository, it also stores server process data for activating servers and objects. When a client invokes on an object, the client ORB sends this invocation to the locator daemon, and the locator daemon searches the implementation repository for the address of the server object. In addition, enables servers to be moved from one host to another without disrupting client request processing. Redirects requests to the new location and transparently reconnects clients to the new server instance. See also [location domain](#), [node daemon](#), and [implementation repository](#).

N**naming service**

See [CORBA naming service](#).

node daemon

Starts, monitors, and manages servers on a host machine. Every machine that runs a server must run a node daemon.

O**object reference**

Uniquely identifies a local or remote object instance. Can be stored in a CORBA naming service, in a file or in a URL. The contact details that a client application uses to communicate with a CORBA object. Also known as interoperable object reference (IOR) or proxy.

OMG

Object Management Group. An open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications, including CORBA. See www.omg.com.

ORB

Object Request Broker. Manages the interaction between clients and servers, using the Internet Inter-ORB Protocol (IIOP). Enables clients to make requests and receive replies from servers in a distributed computer environment. Key component in CORBA.

OTS

See [CORBA transaction service](#).

P**POA**

Portable Object Adapter. Maps object references to their concrete implementations in a server. Creates and manages object references to all objects used by an application, manages object state, and provides the infrastructure to support persistent objects and the portability of object implementations between different ORB products. Can be transient or persistent.

protocol

Format for the layout of messages sent over a network.

S**server**

A program that provides services to clients. CORBA servers act as containers for CORBA objects, allowing clients to access those objects using IDL interfaces.

SSL

Secure Sockets Layer protocol. Provides transport layer security—authenticity, integrity, and confidentiality—for authenticated and encrypted communications between clients and servers. Runs above TCP/IP and below application protocols such as HTTP and IIOP.

SSL handshake

An SSL session begins with an exchange of messages known as the SSL handshake. Allows a server to authenticate itself to the client using public-key encryption. Enables the client and the server to co-operate in the creation of symmetric keys that are used for rapid encryption, decryption, and tamper detection during the session that follows. Optionally, the handshake also allows the client to authenticate itself to the server. This is known as mutual authentication.

T**TCP/IP**

Transmission Control Protocol/Internet Protocol. The basic suite of protocols used to connect hosts to the Internet, intranets, and extranets.

TLS

Transport Layer Security. An IETF open standard that is based on, and is the successor to, SSL. Provides transport-layer security for secure communications. See also [SSL](#).

Index

A

- active connection management
 - HTTP 67
 - IIOp 74
 - SHMIOP 144
- agent_ior_file 119
- AJP policies
 - buffer sizes
 - maximum 158
- AJP policy
 - ports 158
- allow_registration_after_rollback_only 119
- ATL12 46
- AutomaticWorkQueue 34

B

- backoff_ratio
 - binding 159
 - reinvoking 174
- backup_restart_file 119
- Baltimore toolkit
 - selecting for C++ applications 193
- BiDirPolicy::ALLOW 162
- BiDirPolicy::BiDirAcceptPolicy 162
- BiDirPolicy::BiDirExportPolicy 162
- BiDirPolicy::BiDirOfferPolicy 163
- binding:client_binding_list 23
- binding:server_binding_list 24
- binding:servlet_binding_list 25
- binding policies 159
 - forwarding limit 160
 - initial retry delay 159
 - retry delay 159
 - retry maximum 160
 - timeout 160
 - transparent retries 175
- bindings
 - client-side 23
 - server-side 24
- buffer 26
- buffer:heap_storage_size 26

C

- callbacks 164
- CertConstraintsPolicy 191
- CertConstraintsPolicy policy 191
- certificate_constraints_policy variable 191
- Certificates
 - constraints 191
- certificates
 - CertConstraintsPolicy policy 191
 - constraint language 191
- checkpoint_archive_old_files 105, 148
- checkpoint_archives_old_logs 132
- checkpoint_deletes_old_logs 105, 132, 149
- checkpoint_interval 106, 132, 149
- checkpoint_min_size 106, 132, 149
- checkpoints
 - log for PSS 139
- CIO 46
- classloader:cache_scrub_time 40
- classloader:cache_url 38
- classloader:force_explode_wars_to_disk 39
- classloader:jarcache_low_watermark 38
- classloader:jarchache_high_watermark 38
- classloader:jar_dependency_list 40
- classloader:use_single_classloader 39
- classloader:use_single_classloader_for_webinf 39
- client_binding_list 23, 48
- client_version_policy
 - EGMIOP 161
 - IIOp 169, 230
 - SHMIOP 176
- close_string 246
- COMet:config:COMET_SHUTDOWN_POLICY 10
- COMet:config:SINGLE_THREAD_CALLBACK 10
- COMet:debug:MessageLevel 13
- COMet:mapping:KEYWORDS 12
- COMet:mapping:SAFEARRAYS_CONTAIN_VARIANT_S 12
- COMet:services:NameService 18
- COMet:TypeMan:TYPEMAN_CACHE_FILE 14
- COMet:TypeMan:TYPEMAN_DISK_CACHE_SIZE 14
- COMet:TypeMan:TYPEMAN_IFR_NS_NAME 16

- COMet:TypeMan:YPEMAN_IOR_FILENAME 15
- COMet:TypeMan:YPEMAN_LOG_FILE 16
- COMet:TypeMan:YPEMAN_LOGGING 16
- COMet:TypeMan:YPEMAN_MEM_CACHE_SIZE 17
- COMet:TypeMan:YPEMAN_READONLY 17
- COMet configuration
 - cache file location 14
 - callback processing 10
 - disk cache size 14
 - log file output 16
 - log messages 13
 - log output 16
 - memory cache size 17
 - naming service 18
 - SafeArray mapping 12
 - shutdown policy 10
 - switch interface repository 15
- COMET_SHUTDOWN_POLICY 10
- compression 152
- concurrent_transaction_map_size 111
- concurrent_users 132
- configuration:domain_dir 42
- configuration directory
 - path specified in configuration 42
- configuration domain
 - name specified in configuration 42
- configuration variables
 - application level security
 - default_domain 87
 - domain_classname 87
 - domain_list 87
 - file_list 88
 - file_name 88
 - init_at_startup_list 87
 - classloading 38
 - cache_scrub_time 40
 - cache_url 38
 - jarcache_high_watermark 38
 - jarcache_low_watermark 38
 - jar_dependency_list 40
 - use_single_classloader 39
 - use_single_classloader_for_webinf 39
 - data type 4
 - constructed 4
 - names and ports
 - http_addr_list 177
 - https_addr_list 177
- connection_attempts 171, 230
- constraint language 191

- Constraints
 - for certificates 191
- create_dirs 133
- create_transaction_mbeans 119

D

- data_dir 106, 133, 149
- db_home 106, 133, 149
- deadlock detector 139
 - abort 133
 - PSS log 139
- deadlock_detector_aborts 133
- decompression 152, 178
- default_buffer_size 158, 167, 171
- default_ots_policy 112
- default_transaction_policy 112
- default_transaction_timeout 112
- destinations:queue_list 182
- destinations:topic_list 182
- direct_persistence 120
 - event 58
 - IFR 78
 - JMS 188
 - naming service 96
 - notification service 102
 - OTS Encina 120
 - telecom log service 145
- dispatch_strategy 101
- dispatch_threads 102
- domain_dir 42
- domain_name 42

E

- EGMIOP policies
 - client version 161
 - GIOP version in profiles 161
- enable_recovery 89
- event_log:filters 28
- event_pull_interval 59
- event_queue 102
- events_per_transaction 102

F

- factory:password 183
- factory:user 183
- filename 91

G

GIOP
 interoperability policies 164
 policies 164
 giop_snoop 65
 global_namespace_poa 120

H

hard_limit
 HTTP 67
 IIOp 74, 75
 SHMIOP 144
 hard_limit
 HTTP 68
 heap_storage_size 26
 high_water_mark 32
 host, moving to a new 93
 HTTP plug-in configuration
 hard connection limit
 server 67
 harf connection limit
 client 68
 soft connection limit
 client 68
 server 68
 HTTP policies
 buffer sizes
 maximum 167
 ports 168

I

ignore_message_not_consumed 166
 IIOp plug-in configuration
 buffer pool size
 outgoing messages 74
 COMet configuration, recycle buffer segments 74
 hard connection limit
 client 75
 server 74
 number of preallocated buffer segments 74
 soft connection limit
 client 75
 server 74
 IIOp plugin configuration 73
 IIOp policies 169, 223, 228
 buffer sizes 171
 default 171
 maximum 171

client version 169, 230
 connection attempts 171, 230
 export hostnames 169, 172, 235
 export IP addresses 169, 172, 235
 GIOP version in profiles 172, 235
 server hostname 171, 234
 TCP options
 delay connections 172, 236
 receive buffer size 173, 237
 IIOp policy
 ports 172, 234
 initial_disk 120
 initial_disk_size 120
 initial_iteration_delay, binding 159
 initial_reference:IT_JMSMessageBroker:reference 2
 1
 initial_reference:IT_JMSServer:reference 21
 initial_reference:TransactionCurrent:plugin 22
 initial_reference:TransactionFactory:reference 22
 initial_reference:TransactionManager:plugin 22
 initial_reference:UserTransaction:plugin 22
 initial references
 Encina transaction factory 123
 OTS lite transaction factory 117
 OTS transaction factory 115
 specify in configuration 20
 transaction factory 22
 initial_references 20
 initial_references:IT_CSI:plugin 21
 initial_threads 32
 init_txn 134
 instrumentation:enabled 184
 interceptors
 client request-level 23
 interoperability configuration 164
 code set negotiation 166
 GIOP 1.1 support 164
 incompatible message format 166
 LocateRequest messages 166
 Principal data 166
 interposition_style 112
 invocation policies 174
 forwarding limit 174
 initial retry delay 174
 retry delay 174
 retry maximum 175
 ip:receive_buffer_size 56, 68, 75
 ip:send_buffer_size 56, 68, 74
 IT_CodeSet_Registry:plugin 20

J

Java CIO 46
 Java NIO 46
 Java Transaction API. *See* JTA
 JCE architecture
 enabling 194
 jit_transactions 114
 jmx:adaptor:enabled 185
 jmx:adaptor:port 185
 JTA plug-in configuration
 persistent POA 89
 recovery 89
 JTA plugin configuration 89
 plug-in configuration variables 89

K

KEYWORDS 12

L

lb_default_initial_load 97
 lb_default_load_timeout 97
 lk_max 106, 150
 local_hostname 171, 234
 local_log_stream plugin configuration 91
 location_domain_name 94
 locator daemon configuration 93
 IIOP/TLS port 94
 IIOP port 94
 location domain name 94
 NT service dependencies 95
 lock waits, log for PSS 140
 log_check_interval 121
 log_dir
 notification service 106
 PSS 135
 telecom logservice 149
 logging configuration
 set filters for subsystems 28
 logstream configuration
 output stream 91
 output to local file 91
 output to rolling file 92
 log_threshold 120
 low_water_mark 33

M

ManualWorkQueue 34
 max_binding_iterations 160

max_buffer_size 158, 167, 171
 max_forwards
 binding 160
 reinvoking 174
 max_outgoing_buffers_in_pool 74
 max_proxy_consumer_retries 59
 max_proxy_retries 59
 max_proxy_supplier_retries 59
 max_queue_length 59
 max_queue_size 33
 max_rebinds 175
 max_resource_failures 121
 max_retries 107, 150, 175
 max_sleep_time 107, 150
 MessageLevel 13
 message-level interceptors 23
 multi-homed hosts
 clients 169
 servers 171
 multi-homed hosts, configure support for 234

N

NameService 18
 namespace
 binding 23
 classloader 38
 COMet 9
 COMet:config 10
 COMet:debug 13
 COMet:mapping 12
 COMet:services 18
 COMet:TypeMan 14
 configuration 41
 destinations 182
 domian_plugins 27
 event_log 28
 factory 183
 initial_references 20
 instrumentaiton 184
 jmx:adaptor 185
 orb_management 29
 persistence 186
 plugins:ajp 46
 plugins:atli2_ip 46
 plugins:atli2_shm 47
 plugins:basic_log 49, 62
 plugins:codeset 50
 plugins:csi 195
 plugins:egmiop 56

- plugins:event 58
 - plugins:file_security_domain 88
 - plugins:gsp 196
 - plugins:http 67
 - plugins:https 67
 - plugins:i18n 71
 - plugins:ifr 78
 - plugins:iiop 73
 - plugins:iiop_tls 78
 - plugins:iiop_tls:incoming_connections 78
 - plugins:it_http_sessions 80
 - plugins:it_mgmt 81
 - plugins:it_pluggable_http_sessions 83
 - plugins:it_response_time_collector 85
 - plugins:it_security_service 87
 - plugins:jms 188
 - plugins:notify_log 109
 - plugins:ots_mgmt 125
 - plugins:poa 127
 - plugins:pss 128
 - plugins:shmiop 144
 - plugins:ziop 152
 - poa:fqpn 30
 - policies 154, 156, 157, 214
 - policies:binding_establishment 159
 - policies:csi 220
 - policies:egmiop 161
 - policies:giop:interop 161
 - policies:http 167
 - policies:https 223
 - policies:iiop 169
 - policies:iiop_tls 227
 - policies:invocation_retry 174
 - policies:shmiop 176
 - policies:ziop 178
 - principal_sponsor:csi 242
 - principle_sponsor 238
 - root 7
 - thread_pool 32
 - url_resolvers 35
 - namespace_poa 121
 - naming service configuration 96
 - default initial load value 97
 - default load value timeout 97
 - NT service dependencies 97
 - negotiate_transmission_codeset 166
 - new I/O 46
 - NIO 46
 - node daemon configuration 99
 - IIOp/TLS port 100
 - IIOp port 99
 - no_delay 172, 236
 - non_tx_target_policy 154
 - notification service configuration 101
 - database behavior 105, 148
 - event queuing 102
 - events per transaction 102
 - log database events 103
 - logging 103
 - threads available 102
 - thread strategy 101
 - nt_service_dependencies 95, 97
- O**
- old_log_dir
 - notification service 107
 - PSS 136
 - telecom log service 150
 - open_string 246
 - operation_timeout_interval 59
 - orb_management:retrieve_existing_orb 29
 - orb_name
 - OTS Encina 121
 - OTS Lite 116
 - orb_plugins 7
 - otid_format_id
 - OTS Encina 121
 - OTS Lite 116
 - OTS configuration 111
 - default timeout 112
 - hash table size 111
 - initial reference for factory 115
 - initial reference for transaction factory 115
 - interposition style 112
 - JIT transaction creation 114
 - optimize transaction propagation 114
 - OTSPolicy default value 112
 - roll back transactions 114
 - TransactionPolicy default 112
 - transaction timeout default 112
 - OTS Encina configuration 118
 - backup restart file 119
 - direct persistence 120
 - initial log file 120
 - internal ORB usage 123
 - log file growth checks 121
 - log file size 120
 - log file threshold 120

- logging configuration 122
 - log resource failures 121
 - management agent IOR 119
 - ORB name 121
 - OTS management object creation 119
 - POA namespace 121
 - raw disk usage 124
 - registration after rollback 119
 - restart file 122
 - retry timeout 121
 - transaction factory initial reference 123
 - transaction factory name 123
 - transaction ID 121
 - transaction timeout 123
 - OTS Lite configuration 116
 - internal ORB 117
 - ORB name 116
 - transaction ID 116
 - transaction timeout 117
 - OTSMangement:plugin 22
 - OTS management configuration 125
 - enabled 125
 - JMX usage 125
 - manager name 126
 - object creation enabled 125
 - port number 126
 - ots_v11_policy 114
- P**
- persistence:jdbc:connection_pool:max 187
 - persistence:jdbc:connection_pool:min 187
 - persistence:jdbc:driver 186
 - persistence:jdbc:max_message_size 187
 - persistence:jdbc:password 187
 - persistence:jdbc:url 186
 - persistence:jdbc:user 186
 - persistence:message_store 186
 - ping_period 246
 - plug-ins
 - specify in configuration 43
 - transaction factory 22
 - plugins
 - iiop
 - buffer_pools
 - recycle_segments 74
 - segment_preallocation 74
 - pool
 - java_max_threads 75
 - java_min_threads 76
 - loaded on ORB initialization 7
 - OTS management service 22
 - plugins:atli2_ip:ClassName 46
 - plugins:atli2_ip:nio:allocate_heap_byte_buffer 46
 - plugins:atli2_shm:max_buffer_wait_time 47
 - plugins:atli2_shm:shared_memory_segment 48
 - plugins:atli2_shm:shared_memory_segment_basena
 - me 47
 - plugins:atli2_shm:shared_memory_size 48
 - plugins:basic_log:is_managed 49
 - plugins:basic_log:shlib_name 49
 - plugins:codeset:always_use_default 50, 56
 - plugins:codeset:char:ccs 51
 - plugins:codeset:char:ncs 51
 - plugins:codeset:interop_allow_null_strings 50
 - plugins:codeset:wchar:ncs 52
 - plugins:codesets:wchar:ccs 53
 - plugins:config_rep:refresh_master_interval 54
 - plugins:csi:ClassName 195
 - plugins:csi:shlib_name 195
 - plugins:event_log
 - is_managed 62
 - plugins:event_log:shlib_name 62
 - plugins:file_security_domain 88
 - plugins:file_security_domain:file_list 88
 - plugins:file_security_domain:file_name 88
 - plugins:giop:message_server_binding_list 63
 - plugins:giop_snoop:ClassName 64
 - plugins:giop_snoop:filename 65
 - plugins:giop_snoop:rolling_file 65
 - plugins:giop_snoop:shlib_name 66
 - plugins:giop_snoop:verbosity 66
 - plugins:gsp:authorization_realm 198
 - plugins:gsp:ClassName 198
 - plugins:http:connection
 - max_unsent_data 67
 - plugins:http:incoming_connections:hard_limit 67
 - plugins:http:incoming_connections:soft_limit 68
 - plugins:http:outgoing_connections:soft_limit 68
 - plugins:http:tcp_connection:keep_alive 69
 - plugins:http:tcp_connection:linger_on_close 70
 - plugins:http:tcp_connection:no_delay 69
 - plugins:http:tcp_listener:reincarnate_attempts 70
 - plugins:i18n:characterencoding:ianacharset-javacon
 - vector-map 71
 - plugins:i18n:characterencoding:url-inputcharset-ma

- p 71
- plugins:i18n:locale:locale-iancharset-map 72
- plugins:ifr:direct-persistence 78
- plugins:ifr:iioport:host 79
- plugins:ifr:iioport:port 78
- plugins:iioport:buffer_pools:max_outgoing_buffers_in_pool 74
- plugins:iioport:buffer_pools:recycle_segments 74
- plugins:iioport:buffer_pools:segment_preallocation 74
- plugins:iioport:connection_max_unsent_data 74
- plugins:iioport:incoming_connections:hard_limit 74
- plugins:iioport:incoming_connections:soft_limit 74
- plugins:iioport:ip:receive_buffer_size 75
- plugins:iioport:ip:reuse_addr 75
- plugins:iioport:ip:send_buffer_size 74
- plugins:iioport:outgoing_connections:hard_limit 75
- plugins:iioport:outgoing_connections:soft_limit 75
- plugins:iioport:pool:max_threads 76
- plugins:iioport:pool:min_threads 76
- plugins:iioport:tcp_connection:keep_alive 76
- plugins:iioport:tcp_connection:linger_on_close 77
- plugins:iioport:tcp_connection:no_delay 76
- plugins:iioport:tcp_connection:no_delay 76
- plugins:iioport:tcp_connection:linger_on_close 77
- plugins:iioport:tcp_listener:reincarnation_attempts 77, 206
- plugins:iioport:tcp_listener:reincarnation_retry_backoff_ratio 77, 206
- plugins:iioport:tcp_listener:reincarnation_retry_delay 77, 206
- plugins:iioport_tls:hfs_keyring_filename 207
- plugins:iioport_tls:hfs_keyring_file_password 231
- plugins:iioport_tls:hfs_keyring_file_stashfile 207
- plugins:iioport_tls:racf_keyring 207
- plugins:iioport_tls:tcp_listener:reincarnation_retry_backoff_ratio 206
- plugins:iioport_tls:tcp_listener:reincarnation_retry_delay 206
- plugins:it_http_sessions 80
- plugins:it_http_sessions:ClassName 80
- plugins:it_mbean_monitoring:sampling_period 82
- plugins:it_mbean_monitoring:workqueue 82
- plugins:it_pluggable_http_sessions 83
- plugins:it_pluggable_http_sessions:ClassName 83
- plugins:it_pluggable_http_sessions:contexts 83
- plugins:it_pluggable_http_sessions:default_mechanism 84
- plugins:it_pluggable_http_sessions:mechanisms 84
- plugins:it_response_time_collector 85
- plugins:it_response_time_collector:filename 85
- plugins:it_response_time_collector:period 85
- plugins:it_response_time_collector:server-id 86
- plugins:it_response_time_collector:syslog_appID 86
- plugins:it_security_service 87
- plugins:it_security_service:default_domain 87
- plugins:it_security_service:domain_list 87
- plugins:it_security_service:HOSTNAME 87
- plugins:it_security_service:init_at_startup_list 87
- plugins:jms 188
- plugins:jms:direct_persistence 188
- plugins:jms:iioport:port 188
- plugins:jms:is_managed 188
- plugins:local_log_stream:buffer_file 91
- plugins:local_log_stream:filename 91
- plugins:local_log_stream:log_elements 92
- plugins:locator:allow_node_daemon_change 93
- plugins:locator:iioport:port 94
- plugins:locator:iioport_tls:port 94
- plugins:locator:location_domain_name 94
- plugins:locator:node_daemon_heartbeat_interval 94
- plugins:locator:nt_service_dependencies 95
- plugins:locator:refresh_master_interval 95
- plugins:naming:destructive_methods_allowed 96
- plugins:naming:direct_persistence 96
- plugins:naming:iioport:port 97
- plugins:naming:refresh_master_interval 98
- plugins:node_daemon:heartbeat_interval_timeout 99
- plugins:node_daemon:iioport:port 99
- plugins:node_daemon:iioport_tls:port 100
- plugins:node_daemon:is_managed 99
- plugins:node_daemon:recover_processes 100
- plugins:node_daemon:register_interval 100
- plugins:notify:direct_persistence 102
- plugins:notify:iioport:port 103
- plugins:notify_log 111
- plugins:notify_log:is_managed 109
- plugins:notify_log:shlib_name 109
- plugins:ots_encina:iioport:port 120
- plugins:ots_mgmt:create_transaction_mbeans 125
- plugins:ots_mgmt:enabled 125
- plugins:ots_mgmt:jmx_httpd_enabled 125
- plugins:ots_mgmt:jmx_httpd_port 126
- plugins:ots_mgmt:transaction_manager_name 126
- plugins:poa:ClassName 127
- plugins:poa:root_name 127
- plugins:pss:disable_caching 128

- plugins:pss_db:envs:env-name:lg_bsize 134
- plugins:pss_db:envs:env-name:lg_max 134
- plugins:pss_db:envs:env-name:lk_max_lockers 135
- plugins:pss_db:envs:env-name:lk_max_locks 135
- plugins:pss_db:envs:env-name:lk_max_objects 135
- plugins:shmiop:incoming_connections:hard_limit 144
- plugins:shmiop:incoming_connections:soft_limit 144
- plugins:shmiop:outgoing_connections:hard_limit 144
- plugins:shmiop:outgoing_connections:soft_limit 144
- plugins:tlog:direct_persistence 145
- plugins:tlog:flush_interval 145
- plugins:tlog:iioport 145
- plugins:tlog:iterator_timeout 146
- plugins:tlog:max_records 146
- plugins:tlog:trace:events 146
- plugins:tlog:trace:flush 146
- plugins:tlog:trace:lifecycle 146
- plugins:tlog:trace:repository 147
- plugins:ziop
 - shlib_name 152
- plugins:ziop:ClassName 152
- POA
 - plugin class name 127
 - root name 127
- POA::create_POA() 162
- poa:fqpn:direct_persistent 30
- poa:fqpn:well_known_address 30
- poa_name
 - XA plugin 245
- polices:max_chain_length_policy 216
- policies
 - CertConstraintsPolicy 191
- policies:ajp:buffer_sizes_policy:max_buffer_size 158
- policies:ajp:server_address_mode_policy:port_range 158
- policies:allow_unauthenticated_clients_policy 214
- policies:binding_establishment:backoff_ratio 159
- policies:binding_establishment:initial_iteration_delay 159
- policies:binding_establishment:max_binding_iterations 160
- policies:binding_establishment:max_binding_iterations 160
- policies:binding_establishment:max_forwards 160
- policies:binding_establishment:relative_expiry 160
- policies:certificate_constraints_policy 215
- policies:csi:attribute_service:client_supports 220
- policies:csi:attribute_service:target_supports 221
- policies:csi:auth_over_transport:target_supports 222
- policies:csi:auth_over_transport:client_supports 221
- policies:csi:auth_over_transport:target_requires 222
- policies:egmiop:client_version_policy 161
- policies:egmiop:server_version_policy 161
- policies:giop:bidirectional_accept_policy 162
- policies:giop:bidirectional_export_policy 162
- policies:giop:bidirectional_gen3_accept_policy 163
- policies:giop:bidirectional_offer_policy 163
- policies:giop:interop:allow_value_types_in_1_1 164
- policies:giop:interop:cache_is_a 164
- policies:giop:interop:ignore_message_not_consumed 166
- policies:giop:interop:negotiate_transmission_codeset 166
- policies:giop:interop:send_locate_request 166
- policies:giop:interop:send_principal 166
- policies:giop:interop_policy:enable_principal_service_context 165
- policies:http:buffer_sizes_policy:max_buffer_size 167
- policies:http:keep-alive:enabled 167
- policies:http:server_address_mode_policy:port_range 168
- policies:http:transfer-encoding:chunked:enabled 168
- policies:http:transfer-encoding:chunked:reserved_buffer_size 168
- policies:https:allow_unauthenticated_clients_policy 223
- policies:https:certificate_constraints_policy 223
- policies:https:client_secure_invocation_policy:requires 224
- policies:https:client_secure_invocation_policy:supports 224
- policies:https:max_chain_length_policy 224
- policies:https:mechanism_policy:ciphersuites 225
- policies:https:mechanism_policy:protocol_version 225
- policies:https:session_caching_policy 226
- policies:https:target_secure_invocation_policy:requires 226
- policies:https:target_secure_invocation_policy:supports 226

- policies:https:trusted_ca_list_policy 227
- policies:iiop:buffer_sizes_policy:default_buffer_size 171
- policies:iiop:buffer_sizes_policy:max_buffer_size 171
- policies:iiop:client_address_mode_policy:local_hostname 170
- policies:iiop:client_address_mode_policy:port_range 170
- policies:iiop:client_version_policy 169
- policies:iiop:connection_attempts 171
- policies:iiop:server_address_mode_policy:local_hostname 171
- policies:iiop:server_address_mode_policy:port_range 172
- policies:iiop:server_address_mode_policy:publish_hostname 169, 172
- policies:iiop:server_version_policy 172
- policies:iiop:tcp_options:send_buffer_size 173
- policies:iiop:tcp_options_policy:no_delay 172
- policies:iiop:tcp_options_policy:recv_buffer_size 173
- policies:iiop_tls:allow_unauthenticated_clients_policy 229
- policies:iiop_tls:certificate_constraints_policy 229
- policies:iiop_tls:client_secure_invocation_policy:requires 230
- policies:iiop_tls:client_secure_invocation_policy:supports 230
- policies:iiop_tls:client_version_policy 230
- policies:iiop_tls:connection_attempts 230
- policies:iiop_tls:connection_retry_delay 231
- policies:iiop_tls:max_chain_length_policy 231
- policies:iiop_tls:mechanism_policy:ciphersuites 232
- policies:iiop_tls:mechanism_policy:protocol_version 232
- policies:iiop_tls:server_address_mode_policy:local_hostname 234
- policies:iiop_tls:server_address_mode_policy:port_range 234
- policies:iiop_tls:server_address_mode_policy:publish_hostname 235
- policies:iiop_tls:server_version_policy 235
- policies:iiop_tls:session_caching_policy 235
- policies:iiop_tls:target_secure_invocation_policy:requires 236
- policies:iiop_tls:target_secure_invocation_policy:supports 236
- policies:iiop_tls:tcp_options:send_buffer_size 237
- policies:iiop_tls:tcp_options_policy:no_delay 236
- policies:iiop_tls:tcp_options_policy:recv_buffer_size 237
- policies:iiop_tls:trusted_ca_list_policy 237
- policies:invocation_retry:backoff_ratio 174
- policies:invocation_retry:initial_retry_delay 174
- policies:invocation_retry:max_forwards 174
- policies:invocation_retry:max_rebinds 175
- policies:invocation_retry:max_retries 175
- policies:mechanism_policy:ciphersuites 216
- policies:mechanism_policy:protocol_version 217
- policies:non_tx_target_policy 154
- policies:rebind_policy 154
- policies:relative_binding_exclusive_request_timeout 157
- policies:relative_binding_exclusive_roundtrip_timeout 157
- policies:relative_connection_creation_timeout 157
- policies:relative_request_timeout 156
- policies:relative_roundtrip_timeout 156
- policies:routing_policy_max 154
- policies:routing_policy_min 155
- policies:session_caching_policy 217, 218
- policies:shmiop 176
- policies:shmiop:client_version_policy 176
- policies:shmiop:server_version_policy 176
- policies:sync_scope_policy 155
- policies:target_secure_invocation_policy:requires 218
- policies:target_secure_invocation_policy:supports 218
- policies:trusted_ca_list_policy 219
- policies:well_known_addressing_policy:ajp13:address_list 177
- policies:well_known_addressing_policy:http:address_list 177
- policies:well_known_addressing_policy:https:address_list 177
- policies:work_queue_policy 155
- policies:ziop:compression_enabled 178
- policies:ziop:compression_threshold 179
- policies:ziop:compressor:compressor_id:level 179
- policies:ziop:compressor_id 178
- pool:java_max_threads 56, 68, 75
- pool:java_min_threads 57, 69, 76
- pool:max_threads 57, 69, 76
- pool:min_threads 57, 69, 76
- port
 - locator daemon

- IIOIP 94
- IIOIP/TLS 94
- node daemon
 - IIOIP 99
 - IIOIP/TLS 100
- principal_sponsor:csi:auth_method_data 243
- principal_sponsor:csi:use_principal_sponsor 242
- principal_sponsor Namespace Variables 238
- principle_sponsor:auth_method_data 239
- principle_sponsor:auth_method_id 239
- principle_sponsor:callback_handler:ClassName 241
- principle_sponsor:login_attempts 241
- principle_sponsor:use_principle_sponsor 238
- private
 - notification service 108
 - PSS 138
 - telecom log service 151
- process
 - moving to a new host 93
- propagate_separate_tid_optimization 114
- proxy_consumer_retry_multiplier 60
- proxy_inactivity_timeout 60
- proxy interposition 113
- proxy_reap_frequency 60
- proxy_retry_delay 60
- proxy_retry_multiplier 61
- proxy_supplier_retry_delay 61
- proxy_supplier_retry_multiplier 61
- PSS configuration 128
 - Berkeley DB database home directory 133
 - caching 128
 - checkpoint interval 132
 - checkpoint size minimum 132
 - database file name 141
 - data storage directory 133
 - deadlock detector 139
 - abort 133
 - directory creation 133
 - fatal recovery 138
 - logging
 - all events 139
 - archive old files 132
 - checkpoints 139
 - deadlock detection 139
 - delete old files 132
 - lock waits 140
 - log file directory 135
 - old log file directory 136
 - recovery 140

- maximum concurrent PSS transactions 139
- storage home configuration 141
 - See also* storage home configuration
- temporary files directory 139
- thread usage 132
- transaction usage 134
- verbosity 139
- publish_hostname 172, 235

R

- rebind_policy 154
- recover_fatal 108, 138, 151
- recovery
 - log for PSS 140
- recv_buffer_size 173, 237
- recycle_segments 74
- relative_binding_exclusive_request_timeout 157
- relative_binding_exclusive_roundtrip_timeout 157
- relative_connection_creation_timeout 157
- relative_expiry 160
- relative_request_timeout 156
- relative_roundtrip_timeout 156
- resource_poa_name 89
- resource_retry_timeout 121
- restart_file 122
- retrieve_existing_orb 29
- rmid 246
- rollback_only_on_system_ex 114
- rolling_file 92
- root namespace 7
 - orb_plugins 7
 - secure_directories 8
- routing_policy_max 154
- routing_policy_min 155
- run_deadlock_detector 139

S

- SAFEARRAYS_CONTAIN_VARIANTS 12
- Schannel toolkit
 - selecting for C++ applications 193
- secure_directories 8
- segment_preallocation 74
- send_locate_request 166
- send_principal 166
- server_binding_list 24
- server ID, configuring 86
- server process
 - moving to a new host 93

- server_version_policy
 - EGMIOP 161
 - IIOp 172, 235
 - SHMIOP 176
 - shared 48
 - SHMIOP plug-in configuration
 - hard connection limit
 - client 144
 - server 144
 - soft connection limit
 - client 144
 - server 144
 - SHMIOP plugin configuration 144
 - SHMIOP policies 176
 - client version 176
 - server version 176
 - simple_persistent demo 30
 - SINGLE_THREADED_CALLBACK 10
 - soft_limit
 - HTTP 68
 - IIOp 74, 75
 - SHMIOP 144
 - SO_REUSEADDR 75
 - SSL/TLS
 - selecting a toolkit, C++ 193
 - stack_size 34
 - standard interposition 113
 - storage home configuration
 - binary tree keys 142
 - binary tree usage 142
 - cache size 142
 - database cache size 143
 - file creation 142
 - file mode 142
 - file name 141
 - hash table density 143
 - hash table size 143
 - page size 143
 - read only 142
 - truncate file 142
 - superior_ping_timeout 116
 - support_ots_v11 115
 - supports_async_rollback 246
 - sync_scope_policy 155
 - sync_transactions 108, 151
- T**
- t 61
 - TCP policies
 - delay connections 172, 236
 - receive buffer size 173, 237
 - telecom log service configuration
 - log database events 146
 - thread_pool:high_water_mark 32
 - thread_pool:initial_threads 32
 - thread_pool:low_water_mark 33
 - thread_pool:max 33
 - thread_pool:max_queue_size 33
 - thread pool policies 32
 - initial number of threads 32
 - maximum threads 32
 - minimum threads 33
 - request queue limit 33
 - timeout policies 156
 - tmp_dir
 - notification service 108
 - PSS 139
 - telecom log service 151
 - toolkit replaceability
 - enabling JCE architecture 194
 - selecting the toolkit, C++ 193
 - trace:database 103, 146
 - trace:events 61, 103
 - trace:filters 103
 - trace:lifecycle 61, 103
 - trace:locks 103, 147
 - trace:queue 104
 - trace:retry 104
 - trace:subscription 104
 - trace:transactions 104, 147
 - trace_file 122
 - trace_on 123
 - transaction factory, initial reference 115
 - TransactionFactory:plugin 22
 - transaction_factory_name
 - OTS 115
 - OTS Encina 123
 - OTS Lite 117
 - transaction_factory_ns_name 123
 - TransactionPolicy, configure default value 112
 - transactions
 - checkpoint size minimum 132
 - handle non-transactional objects 154
 - log file archiving 132
 - log file deletion 132
 - maximum concurrent in PSS 139
 - usage against database 134
 - transaction_timeout_period

INDEX

- OTS Encina 123
- OTS Lite 117
- ts2idl 12
- tx_max 107, 150
- TYPEMAN_CACHE_FILE 14
- TYPEMAN_DISK_CACHE_SIZE 14
- TYPEMAN_IFR_IOR_FILENAME 15
- TYPEMAN_IFR_NS_NAME 16
- TYPEMAN_LOG_FILE 16
- TYPEMAN_LOGGING 16
- TYPEMAN_MEM_CACHE_SIZE 17
- TYPEMAN_READONLY 17

U

- URLs, configure resolution 35
- use_internal_orb 117, 123
- use_jsse_tk configuration variable 194
- use_raw_disk 124

V

- verb_all 139
- verb_checkpoint 139
- verb_deadlock 139
- verb_recovery 140
- verb_waitsfor 140

W

- work_queue_policy 155

X

- XA plug-in configuration
 - asynchronous rollbacks 246
 - close string default 246
 - open string default 246
 - ping interval 246
 - POA name 245
 - resource manager ID 246
- XA plugin configuration 245

Z

- ziop
 - plug-in 152
 - policies 178

