Progress®



Configure OpenEdge Components with OpenEdge Management

OpenEdge®

Table of Contents

Preface....11

Introduction....13

OpenEdge Management and OpenEdge Explorer....13
OpenEdge Management....14
OpenEdge Explorer....14
Get started....15
OpenEdge Management and OpenEdge Explorer features and licensing....15

Configure OpenEdge Databases....19

Database configuration and administration....20
View the database log file....21
Database configuration awareness....21
Manage databases....21
View or modify database properties....22
Database properties....22
View the various database configuration properties....23
Database configuration properties....23
Edit the database default configuration....38
Create a database configuration....39
Assign a different default database configuration....39
View the server group....40
Database server group properties....40
Edit a database server group....43
Create a database server group....43
Start or stop the database....44
Start a database automatically....44
Background writers and watchdog processes....44
Background writers....45
The watchdog process....45
View database status....45
View database process details....46
Delete a database, database configuration, or database server group....47
Delete a database....47
Delete a database configuration....47
Delete a database server group....48
Configure an OpenEdge Replication-enabled database....48
Rules for setting properties for a Replication-enabled database....48
Set Replication-related properties....49

Set OpenEdge Replication Agent properties....49
Set OpenEdge Replication Server properties....50
Set OpenEdge Replication Control Agent properties....52
Set OpenEdge Replication Transition properties....53

Configure DataServers....57

Oracle DataServer configuration and administration....57
Use the Oracle DataServer broker....58
Configure an Oracle DataServer broker....58
Edit an Oracle DataServer broker configuration....65
Create an Oracle DataServer broker....65
Delete an Oracle DataServer broker....65
Start or stop an Oracle DataServer broker....66
Start an Oracle DataServer broker automatically....66
View the status of an Oracle DataServer broker....66
Oracle DataServer broker property Status summary and details....67
View the Oracle DataServer broker log files....68
MS SQL Server DataServer configuration and administration....69
Use the MS SQL Server DataServer broker....69
Configure an MS SQL Server DataServer broker....70
Edit an MS SQL Server DataServer broker configuration....76
Create an MS SQL Server DataServer broker....76
Delete an MS SQL Server DataServer broker....77
Start or stop an MS SQL Server DataServer broker....77
Start an MS SQL Server DataServer broker automatically....77
View the status of an MS SQL Server DataServer broker....78
View the MS SQL Server DataServer log files....79

Configure NameServers....81

NameServer configuration and administration....82
NameServer features....82
Use with the NameServer....82
NameServer as an optional client connection request handler....83
Local and remote NameServers....83
Configure a NameServer....83
Configure local instances....83
NameServer properties....84
Configure remote NameServer instances....89
Edit a NameServer configuration....90
Set NameServer environment variables....90
Delete NameServer environment variables....90
Create a NameServer....91
Delete a NameServer....91

Configure fault tolerance and load balancing....92
Connection-level fault tolerance....92
Server-level fault tolerance 93
Load balancing....93
Location transparency....93
Start or stop a NameServer....93
Start a NameServer automatically....93
View the status of a NameServer....94
NameServer status summary properties....94
NameServer status detail properties....95
Handle UDP and firewall issues....95
View the NameServer log file....96

Configure AppServers and AppServer Internet Adapters....97

AppServer configuration and administration....97
Overview of AppServer configuration....98
AppServer features....98
AppServer actions....98
AppServer configuration properties....99
Configure an AppServer....100
Edit an AppServer configuration....119
Create an AppServer....119
Start and stop an AppServer....119
Add or trim AppServer agents....120
View the status of an AppServer....121
View AppServer process details....124
Stop an AppServer agent....125
Kill an AppServer agent....125
Delete an AppServer....125
View AppServer log files....126
For additional AppServer information....126
AppServer Internet Adapter configuration and administration....126
Use the AppServer Internet Adapter....127
Configure an AppServer Internet Adapter....127
AppServer Internet Adapter (AIA) properties....128
Edit an AppServer Internet Adapter configuration....133
Create an AppServer Internet Adapter....133
View the AppServer Internet Adapter log file....134
Enable or disable an AppServer Internet Adapter broker....135
Delete an AppServer Internet Adapter broker....135

Configure WebSpeed Messengers and WebSpeed Transaction Servers 137

WebSpeed Messenger administration 137
Configure a WebSpeed Messenger....138
Edit a WebSpeed Messenger configuration....142
View the WebSpeed Messenger log file....142
WebSpeed Transaction Server administration....143
Use the WebSpeed Transaction Server....144
Configure a WebSpeed Transaction Server....144
Edit a WebSpeed Transaction Server configuration....163
Create a WebSpeed broker....163
Start and stop a WebSpeed broker....163
Add or trim WebSpeed agents....164
Delete a WebSpeed broker....165
Additional WebSpeed information....166

Configure SonicMQ Adapters....167

Configure OE Web Servers....169

Configure and manage OE Web Servers....169
REST Management Agent....170
OE Web Server....170
REST Web Applications....170
Apache Tomcat....170
Install a REST Management Agent into a Java AdminServer....170
Manage REST Web Applications with the OE Web Server....171
Manage OE Web Servers....171
Start and stop Tomcat server....172
Create an OE Web Server instance....173
Configure an OE Web Server instance....174
Edit an OE Web Server instance configuration....175
Set OE Web Server application defaults....176
Deploy a REST Web application (.war file)....176
View the status of an OE Web Server instance....177
View OE Web Server instance statistics....177
Change OE Web Server run-time properties....178
Delete an OE Web Server instance....179
Enable or disable an OE Web Server instance 179
View the OE Web Server log file 179
Work with REST Web Applications....180
Enable or disable a REST Web application for client access....180
View REST Web application statistics....181
Use the REST Web application log file viewer....182
Update a deployed REST Web application....183
TLS support for REST Web Applications....184

Undeploy a REST Web application....184

Configure Batch Programs....187

Overview....188
Create a Batch Program....188
Modify Batch Program configuration....188
Start a Batch Program....189
Stop a Batch Program....189
Delete a Batch Program....190
View and manage Batch Program processes....190
View Batch Program log files....191
Batch Program properties....192
Batch Program General properties....196
Batch Program Startup properties....197
Batch Program Monitoring properties....200
Batch Program Shutdown properties....203
Batch Program Environment variable properties....204
Batch Program default actions....204
PID registration....206
API references....207
GET( ) method (Process)....207
POST( ) method (Process)....208
POST( ) method....209
PUT( ) method (Process)....209
DELETE( ) method (Process)....210
GET( ) method (Processes)....211

Configure third-party Web applications....213

Overview....213
Extension mechanism in OpenEdge Management....214
OpenEdge Management security....214
Requirements for hosting a third-party Web application....215
Host a third-party Web application....215
Implement single sign-on (SSO)....215
Access JSP Standard Tag Libraries....216
Load Java classes....216
Page contributions for a third-party Web application....217
Behavior of OpenEdge Management....217
Register a contributor....218
OpenEdge Management Extension framework....218
Access OpenEdge Management internals....218
Third-party Web application layout....219
OpenEdge Management REST API framework....220

Configure a third-party Web application....221
REST API code example....223
URL mapping....224
Exception handling....224
REST handler method response....225
Method annotations....226
Parameter annotations....227
Sample Web application in OpenEdge Management....230
Use the sample Web application....230
Access the REST API extension....231
REST API for the sample Web application....231
Access resources....232
Compile sample Web application using Ant....232

Index....235

Copyright

© 2021 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, DevReach (and design), Icenium, Inspec, Ipswitch, iMacros, Kendo UI, Kinvey, MessageWay, MOVEit, NativeChat, NativeScript, OpenEdge, Powered by Chef, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, Sitefinity (and design), SpeedScript, Stylus Studio, Stylized Design (Arrow/3D Box logo), Styleized Design (C Chef logo), Stylized Design of Samurai, TeamPulse, Telerik, Telerik (and design), Test Studio, WebSpeed, WhatsConfigured, WhatsConnected, WhatsUp, and WS_FTP are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries.

Analytics360, AppServer, BusinessEdge, Chef Automate, Chef Compliance, Chef Desktop, Chef Habitat, Chef WorkStation, Corticon.js, Corticon Rules, Data Access, DataDirect Autonomous REST Connector, DataDirect Spy, DevCraft, Fiddler, Fiddler Everywhere, FiddlerCap, FiddlerCore, FiddlerScript, Hybrid Data Pipeline, iMail, JustAssembly, JustDecompile, JustMock, KendoReact, NativeScript Sidekick, OpenAccess, PASOE, Pro2, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, Push Jobs, SafeSpaceVR, Sitefinity Cloud, Sitefinity CMS, Sitefinity Digital Experience Cloud, Sitefinity Feather, Sitefinity Insight, Sitefinity Thunder, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Supermarket, SupportLink, Unite UX, and WebClient are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

July 2021

Product version: Progress OpenEdge 12.4

Preface

Purpose

This guide describes how to establish property and configuration settings for OpenEdge databases, DataServers (for Oracle and MS SQL Server), NameServers, AppServers, AppServer Internet Adapters, REST Web Services, WebSpeed® Transaction Servers and Messengers, and SonicMQ® Adapters in OpenEdge® Management and OpenEdge Explorer. In addition, the guide also provides detail about viewing status and log files.

Audience

This guide is intended for OpenEdge Management and OpenEdge Explorer users as well as OpenEdge Management database and system administrators.

Organization

Introduction on page 13

Introduces OpenEdge Management and OpenEdge Explorer and describes their features.

Configure OpenEdge Databases on page 19

Describes how to establish and modify property and configuration settings for a database, database configuration, or database server group; add a database instance; start and stop a database; use the background writers and watchdog process; view status; view the log file; and delete a database.

Configure DataServers on page 57

Describes how to establish and modify property and configuration settings for an Oracle or MS SQL Server DataServer; add a DataServer instance; start and stop a DataServer; view status; view the log files; and delete a DataServer.

Configure NameServers on page 81

Describes how to establish and modify property and configuration settings for a NameServer; add a NameServer instance; start and stop a NameServer; work with fault tolerance and load balancing; view status; view the log files; and delete a NameServer.

Configure AppServers and AppServer Internet Adapters on page 97

Describes how to establish and modify property and configuration settings for an AppServer or AppServer Internet Adapter; add an AppServer or AppServer Internet Adapter instance; add or trim AppServer agents; start and stop an AppServer; enable or disable an AppServer Internet Adapter; view status; view the log files; and delete an AppServer or AppServer Internet Adapter.

Configuring OpenEdge SOAP Web Services

Describes how to configure and manage SOAP Web services; and work with SOAP Web services.

Configure WebSpeed Messengers and WebSpeed Transaction Servers on page 137

Describes how to establish and modify property and configuration settings for a WebSpeed Transaction Server or WebSpeed Messenger; create or delete a WebSpeed Transaction Server instance; start and stop a WebSpeed Transaction Server; add or trim WebSpeed Transaction Server agents; view status; and view the log files.

Configure SonicMQ Adapters on page 167

Describes how to establish and modify property and configuration settings for a SonicMQ Adapter; create or delete a SonicMQ Adapter instance; start and stop a SonicMQ Adapter; view status; and view the log file.

Configure OE Web Servers on page 169

Describes how to configure and manage OE Web Servers, REST Web services; manage REST Web Services instances; and work with OE WebServers.

Documentation conventions

See Documentation Conventions for an explanation of the terminology, format, and typographical conventions used throughout the OpenEdge content library.

1

Introduction

OpenEdge® Management is a system management center that provides visibility, analysis, and proactive monitoring of critical information assets. OpenEdge Management optimizes the availability and performance of OpenEdge-based applications through system monitoring, alerting, and automatic handling of corrective actions. OpenEdge Management empowers Progress Software customers to become more efficient, decrease the cost of managing the OpenEdge® environment, and ensure high availability and performance.

OpenEdge Explorer is a subset of OpenEdge Management and runs within the OpenEdge Management console in a Web browser. OpenEdge Explorer enables you to set configuration properties, to start or stop, and to view the status or log files for various OpenEdge resources. OpenEdge Explorer is supported on all the UNIX platforms that support OpenEdge.

For details, see the following topics:

- OpenEdge Management and OpenEdge Explorer
- Get started

OpenEdge Management and OpenEdge Explorer

You can use OpenEdge Management or OpenEdge Explorer to configure your licensed OpenEdge resources.

Note: The AdminServer referenced in the following sections is a background process that provides a common point of management for the resources managed by OpenEdge Management or OpenEdge Explorer. For more information on the AdminServer, see *OpenEdge Getting Started: Installation and Configuration*.

OpenEdge Management

OpenEdge Management provides database administrators and systems operations managers with the performance tools and processes required to configure, monitor, diagnose, and manage the OpenEdge environment. This is all done from an easy-to-use graphical interface known as the *management console*, which runs in a Web browser.

Using OpenEdge Management, you can monitor the following resources running under a local AdminServer or under an AdminServer on a remote machine:

- OpenEdge databases (including those that are enabled for OpenEdge® Replication)
- System resources (CPU, disk, memory, file system)
- File resources
- OpenEdge servers, such as Progress Application Server for OpenEdge (PAS for OpenEdge), Classic AppServer, NameServer, WebSpeed® Transaction Server, and supported DataServers (Oracle and MS SQL Server)
- WebSpeed Messengers
- AppServer Internet Adapters, OE Web Servers, and SonicMQ® Adapters
- TCP-based network services

OpenEdge Management's deep level of monitoring provides information and details about your environment, enabling you to proactively manage operations.

In addition, OpenEdge Management includes the Database Administration Console, which allows you to manage and work with databases.

OpenEdge Explorer

OpenEdge Explorer allows you to set and modify properties from within the OpenEdge Management console. The functionality that you can use for the various resources in OpenEdge Explorer is provided in the following table.

Table 1: OpenEdge Explorer functionality by resource

OpenEdge Resource instance	Set config. props	Edit config. props	Start	Stop	Delete	Create	View generic status	View log file
PAS for OpenEdge	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
AppServer	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
AppServer Internet Adapter	Yes	Yes	No	No	Yes	Yes	Yes	Yes
Database	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Messengers	Yes	Yes	No	No	No	No	Yes	Yes

OpenEdge Resource instance	Set config. props	Edit config. props	Start	Stop	Delete	Create	View generic status	View log file
MSS DataServer	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
NameServer	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Oracle DataServer	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SonicMQ Adapter	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
WebSpeed Transaction Server	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
OE Web Server	Yes	Yes	No	No	Yes	Yes	Yes	Yes

OpenEdge Explorer also includes the Database Administration Console, which allows you to manage and work with databases.

Get started

This guide provides details about how to use OpenEdge Management and OpenEdge Explorer to configure OpenEdge resources.

For introductory information about using the OpenEdge Management and OpenEdge Explorer console, see *OpenEdge Management and OpenEdge Explorer: Getting Started*.

OpenEdge Management and OpenEdge Explorer features and licensing

OpenEdge Management and OpenEdge Explorer are packaged with OpenEdge and do not require a separate installation; however, OpenEdge Management still requires a separate license. When OpenEdge Management is licensed, the OpenEdge Explorer functionality is included. If you did not purchase OpenEdge Management, however, its options are grayed out and unavailable in the management console when you are using OpenEdge Explorer.

In general, the settings you can establish and the options you can access when using OpenEdge Management and OpenEdge Explorer are determined both by the platform you are using and the products that you are licensed to use. If you notice that a particular link is dimmed in the OpenEdge Explorer console and not available for modification, it is because that link has no meaning within your operating environment or your licensed configuration.

Note: The settings (properties) for most of the resources managed by OpenEdge Management and OpenEdge Explorer (except for OpenEdge databases) are stored in the `ubroker.properties` file installed with OpenEdge. These properties are stored as name-value pairs. The names of these properties shown in OpenEdge Management and OpenEdge Explorer are long forms of the short names for the same properties stored in the `ubroker.properties` file. For more information about this file, see *OpenEdge Getting Started: Installation and Configuration*. For information on property storage for OpenEdge databases, see Database configuration and administration on page 20.

Feature comparison

The following table provides a comparison between the functionality available to you if you have purchased and installed OpenEdge Management (with OpenEdge Explorer), and the functionality available to you if you are working solely with OpenEdge Explorer.

Table 2: OEM and OEE feature comparison

Feature	Available in OpenEdge Explorer	Available in OpenEdge Management (with OpenEdge Explorer)
Configuring or modifying default property settings	Yes	Yes
Starting/stopping resource instances (if applicable)	Yes	Yes
Viewing status (if applicable)	Yes	Yes
Viewing log files	Yes	Yes
Adding and configuring new resource instances	Yes	Yes
Supporting resource and log file monitoring for: • Databases • DataServers • AppServers • NameServers • WebSpeed Transaction Servers • Web Speed messengers • Network components (TCP, UDP, PING, HTTP) • System components (CPU, disk, file system, memory) • File components • SonicMQ Adapters • OE Web Servers	No	Yes

Feature	Available in OpenEdge Explorer	Available in OpenEdge Management (with OpenEdge Explorer)
Supporting log file monitoring for: • AppServer Internet Adapters • WebSpeed Messengers	No	Yes
Creating jobs for supported resources	No	Yes
Creating reports for supported resources	No	Yes
Using Library functions, such as rule sets, for supported resources	No	Yes
Alerts for supported resources	No	Yes
Scheduling and polling for supported resources	No	Yes
Graphing for supported resources	No	Yes
Trending for supported resources	No	Yes
SNMP support	No	Yes (with SNMP license)
Dashboard	No	Yes
Collections	No	Yes

2

Configure OpenEdge Databases

You can use OpenEdge Management and OpenEdge Explorer to configure and administer the database configuration of an existing OpenEdge RDBMS.

For information about managing and working with databases enabled for multi-tenancy, see *OpenEdge Management and OpenEdge Explorer: Configuring Multi-tenancy*.

For details, see the following topics:

- Database configuration and administration
- Manage databases
- View or modify database properties
- View the various database configuration properties
- View the server group
- Start or stop the database
- Background writers and watchdog processes
- View database status
- Delete a database, database configuration, or database server group
- Configure an OpenEdge Replication-enabled database

Database configuration and administration

Using OpenEdge Management or OpenEdge Explorer, you can set properties for a database, its various configurations, and its associated server groups. Any configuration settings that you establish for a managed database, database configuration, or database server group in OpenEdge Management or OpenEdge Explorer are automatically reflected in the OpenEdge database Configuration Manager property file (`conmgr.properties`), and vice versa.

Note: You cannot create a new physical database from within OpenEdge Management or OpenEdge Explorer. For more information about creating a database, see *OpenEdge Data Management: Database Administration*.

A database recognized and managed by the AdminServer is called a *managed database*. A database not managed by an AdminServer is called a *scripted database*. Scripted databases are administered outside the AdminServer using parameter files (`.pf`) and operating system-dependent scripts.

A managed OpenEdge database starts up by using the default configuration information stored in the `conmgr.properties` file. The file contains the property settings for all managed databases on the host where OpenEdge is installed.

The `conmgr.properties` file resides in the `properties` subdirectory of the OpenEdge install directory. Each database is represented in the file by the following groups:

- **Database** — The group that describes the physical database. There is one database group for each physical database.
- **Configuration** — One or more groups that describe the database arguments and server groups for an active instance of the database. Only one database configuration can be active at one time. The active configuration is the default configuration identified in the server group.
- **ServerGroup** — One or more groups that describe brokering behavior. The first server group within a configuration defines the primary broker and each subsequent group defines a secondary broker. The properties in a server group define the arguments that are specific to that primary or secondary broker.

You can edit the `conmgr.properties` file manually; any changes that you make to it are automatically reflected in OpenEdge Management or OpenEdge Explorer.

Note: Although you can make manual edits to the `conmgr.properties` file, Progress Software recommends that you use the mergeprop utility, OpenEdge Management, or OpenEdge Explorer to make property changes. For more information about the mergeprop utility, see *OpenEdge Getting Started: Installation and Configuration*.

From OpenEdge Management or OpenEdge Explorer, you can perform the following database-related actions:

- Add an existing database
- Create a new database configuration
- Edit an existing or new database configuration
- Use the default server group
- Create a new database server group
- Edit a new or existing database server group
- Start a database
- Start a database automatically

- Stop a database
- Use the background writers and watchdog processes
- View the status of a database
- Delete a database
- Delete a database configuration (provided there is more than one)
- Delete a database server group (provided there is more than one)
- Configure the properties of an OpenEdge Replication-enabled database

You can also manage and work with databases enabled for multi-tenancy. For details, see *OpenEdge Management and OpenEdge Explorer: Configuring Multi-tenancy*.

View the database log file

You can use the log file viewer to review the database log file. For more information, see the topic about using the log file viewer in *OpenEdge Management: Database Management*.

Database configuration awareness

You can use OpenEdge Management or OpenEdge Explorer to establish database settings and options. These settings and options are determined both by the platform you are using and the products that you are licensed to use. If a property is not applicable to the database, its setting is either ignored or, when appropriate, an error or warning is issued.

Manage databases

You can use OpenEdge Management or OpenEdge Explorer to manage your database. You cannot physically create a database in the management console; however, you can create the definition of a database so that you can configure, start, stop, or view status for the primary broker and all auxiliary processes associated with that broker, and view status for the log file for the database.

Database configuration information is stored in the `conmgr.properties` file. For a complete list of how the database properties map to the command-line `.pf` arguments, see the `conmgr.properties.README`. You can locate the `README` file in the *`OpenEdge-install`*`\properties` directory.

A database recognized and managed by the AdminServer is called a *managed* database. A database not managed by an AdminServer is called a *scripted* database. Scripted databases are administered outside the AdminServer using parameter files (`.pf`) and operating system-dependent scripts. You can migrate a scripted database to a managed database.

For details about adding a managed or a scripted database, see *OpenEdge Management: Database Management*.

For details about managing and working with databases enabled for multi-tenancy, see *OpenEdge Management and OpenEdge Explorer: Configuring Multi-tenancy*.

View or modify database properties

Each database definition has a set of properties. You can view and modify the properties if you want to establish new values.

To view or modify the database properties:

1. Click **Resources** from the OpenEdge Management console menu. The **Resources** page appears with all the resources managed by your console, in a grid list.
2. From the resources list, select a database whose default configuration properties you want to modify.

   Note: You can also search for a specific database by entering the database name in the **Search** field or by selecting Database from the resources type drop-down.

3. Click the **Edit** icon. The **Database** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Database Configuration** page appears. From the **Database Configuration** page, you can do the following:
   - View a read-only display of the database properties as described in Database properties on page 22.
   - Click **Edit** to modify the database properties. When you finish, click **Save**.
   - Click the relevant **Configuration** and **Server Group** link to access the database default configuration properties or the database server group properties. For details about these properties, see View the various database configuration properties on page 23 and View the server group on page 40.
   - Set OpenEdge Replication Server or Agent properties, if the database is replication-enabled or if an associated `<db-name>.repl.properties` file exists in the physical database directory. For details, see Configure an OpenEdge Replication-enabled database on page 48.

Database properties

Table 3: Database properties

Property	Description
Database name	Identifies the full path to the database.
Autostart	Indicates whether the database starts automatically when the AdminServer starts.
Cluster enabled	Indicates whether you have already configured the database as cluster-enabled (by using the PROUTIL DESCRIBE command). This allows the database to register with the cluster manager at start time.
Default configuration	Provides the name of the default configuration that is used when the database starts.

View the various database configuration properties

OpenEdge Management and OpenEdge Explorer provide a predefined default database configuration. Once you add a database instance, you can use the property settings established in the default configuration, you can modify the default settings, or you can create and modify additional configurations.

Note that only one database default configuration can be selected for a database at a time.

To view the database default configuration properties:

1. Click **Resources** from the OpenEdge Management console menu. The **Resources** page appears with all the resources managed by your console, in a grid list.
2. From the resources list, select a database whose default configuration properties you want to view.

   Note: You can also search for a specific database by entering the database name in the **Search** field or selecting Database from the resources type drop-down.

3. Click the **Edit** icon. The **Database** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Database Configuration** page appears.
5. Click **configuration.<*databasename*>.defaultconfiguration**. The **Database Configuration** page appears, displaying the default configuration properties in the following categories: **General**, **Background Writers**, **After Image Management**, **International**, **Advanced**, **Statistics Collection Configuration**, **SQL-92 Configuration**, **SSL Configuration**, **Agent** and **Environment variables**. These properties are described in Database configuration properties on page 23.

From the **Database Configuration** page, you can do the following:

- View a read-only display of the database default configuration properties as described in the tables that follow this procedure.
- Click **Edit** to modify the default database properties. For details, see Edit the database default configuration on page 38.
- Click the relevant link to access the associated database server group properties. For details about these properties, see View the server group on page 40.

Database configuration properties

The following topics describe the database configuration properties.

General database configuration properties

Table 4: General database configuration properties

Property	Description
Blocks in database buffers (primary) (-B)	The number of blocks in the database buffers. The optimum value for this property depends on your application. The default value is 0.

Property	Description
Alternate buffer pool (-B2)	The number of blocks in the alternate buffer pool. The optimum value depends on your application. If you specify **Alternate buffer pool** blocks for a Replication target database, the property setting is ignored. **Note:** Alternate buffer pool functionality requires an Enterprise database license. For more information about alternate buffer pools, see *OpenEdge Data Management: Database Administration*.
Hash table entries (-hash)	The number of hash table entries to use for the buffer pool. The default value is 0. **Note:** Do not edit this property unless directed to do so by Progress Software Technical Support.
Buffer hash latch factor (-hashLatchFactory)	The number of latches allocated for the Buffer pool Hash Table (BHT) entries. The number of latches allocated is calculated based on the percent you specify. The default value is 10.
Lock table entries (-L)	The number of entries in the record locking table. If you specify a value that is not a multiple of 32, the value is rounded to the next highest multiple of 32. Each record that is accessed by any user takes one entry. The default value is 8192.
Tenant Lock Table Governors(%) (-LGovernor)	Specifies the percentage of the lock table that any tenant can occupy. The valid values are 1 through 100. The default value is 100. **Note:** This parameter applies to only multi-tenant databases. In addition, this applies to only the default tenant and regular tenant. The super tenants do not use the governors parameters. You can also monitor this parameter using PROMON utility.
Max users (-n)	The maximum number of users on the system for the database.The valid values are 1 through the maximum number of users (32000) After the maximum number of users have connected to the database, additional user connect attempts are rejected. The default value is 20.

Property	Description
Tenant Max. User Governor (-nGovernor)	Specifies the maximum number of users that may login for any tenant. The valid values are 1 through the maximum number of users (32000). The default value is the maximum number of users (-n). **Note:** This parameter applies to only multi-tenant databases. In addition, this applies to only the default tenant and regular tenant. The super tenants do not use the governors parameters. You can also monitor this parameter using the PROMON utility.
Max servers (-Mn)	The maximum number of remote client servers that can be started by the broker process. Use the **Max servers** property to limit the number of remote user servers that can be started on the system. The performance trade-off to consider is swapping overhead for many servers versus overloading (slowing down) a server with too many clients. **Note:** This parameter pertains only to the OpenEdge® Workgroup RDBMS and the OpenEdge® Enterprise RDBMS. In addition, this parameter applies only to databases that are accessed from remote network nodes.
Maximum private database buffers per user (-Bp)	Controls the number of private buffers a connection can request through the Private Read-only Buffers (-Bp) parameter.
Direct I/O (-directio)	Enables the Direct I/O property, which opens all files in unbuffered mode. Direct I/O enables the database to use an I/O technique that bypasses the operating system buffer pool and transfers data directly from a buffer to disk. This technique has some advantages over buffered reads and writes, such as avoiding the overhead of maintaining the operating system buffer pool, and eliminating competition for operating system buffers between the database and other programs. The operating system buffer-pool algorithms are designed for efficient sequential file access; the OpenEdge buffer-pool algorithms are more efficient for access to an OpenEdge RDBMS. The **Direct I/O** check box is cleared by default.
Other arguments	Any other arguments for which there is no defined property available.

Background writers database configuration properties

Table 5: Background writers database configuration properties

Property	Description
Watchdog process	Start the Watchdog process automatically when the database starts. The check box is selected by default.
Asynchronous page writers	The number of Asynchronous Page Writer (APW) processes to start. A database can have between zero and nine APWs running simultaneously. The optimal number is dependent on your application and environment. To start, use one page writer for each disk where the database resides. If this is insufficient, add more. For an application that performs many updates, start one APW for each disk containing your database, and add one additional APW. Applications that perform fewer changes to a database require fewer APWs. If an application performs no updates, no page writers are required. The default value is 1.
Before image process	Start a Before-imaging process automatically when the database starts. The check box is selected by default.
Before image stall (-bistall)	Send a message to the log file when the recovery log threshold is reached, preventing emergency shutdown. (For details about the `-bistall` startup parameter, see *OpenEdge Data Management: Database Administration*.) The check box is cleared by default.
Before image buffers (-bibufs)	The number of before-image buffers to use. This property is useful only when running the before-image writer (BIW). The BIW continually writes the filled before-image buffers to disk, making the buffers available to other client and server processes. Without a BIW writing the buffers, any gain from increasing the number of buffers is negligible. The default value is 5.
Before image threshold (-bithold)	The maximum size to which BI files can grow as a percentage of the largest possible recovery log file. (For details about the `-bithold` startup parameter, see *OpenEdge Data Management: Database Administration*.) The default value is 3.

Property	Description
Before image delay writes (-Mf)	A positive value that you specify to delay OpenEdge from synchronously writing out to disk the last before-image (BI) file records at the end of each transaction. On UNIX systems using shared memory, the value also specifies the interval at which the broker process wakes up to make sure all BI file changes have been written to disk. The default value is 3.
Before image cluster age (-G)	The number of seconds before OpenEdge reuses a before-image cluster. The default value is 60.
Group delay (-groupdelay)	The number of milliseconds a transaction waits before committing.
After image process	Start an After-imaging process automatically when the database starts. The check box is cleared by default.
After image stall (-aistall)	Suspend database activity if all the after-image (AI) files are filled. By default, if all the AI files are filled and the database cannot switch to an empty one, the database shuts down. With AI stall, you can suspend database activity, back up the full after-image file, and mark it as empty. The check box is cleared by default.
After image buffers (-aibufs)	The number of after-image buffers to use. This parameter is useful only when running the after-image writer (AIW) process because the AIW writes the filled after-image buffers to disk, making the buffers available to other client and server processes. The default value is 5.

After Image Management database configuration properties

Table 6: After Image Management database configuration properties

Property	Description
After-image archival directory list (-aiarcdir)	The directories where the AI File Management utility writes the archived after-image files.

Property	Description
Create after-image archival directories (-aiarcdircreate)	Directs the AI File Management utility to create the specified directories.
After-image archival interval (-aiarcinterval)	Specifies on-demand mode archiving or the extent switch interval for timed-mode archiving. The minimum value is 120, and the maximum value is 86400.The default value is 120.

International database configuration properties

Table 7: International database configuration properties

Property	Description
Collation table (-cpcoll)	A collation table in the `convmap.cp` file that you use for collation rules. The collation table you specify is used with the code page OpenEdge uses in memory, which is specified by the Internal Code Page (`-cpinternal`) parameter. The default value is BASIC.
Case table name (-cpcase)	A case table in the `convmap.cp` file that you use for case rules. Case rules are used by the 4GL CAPS and LC functions and by the exclamation point (!) formatting character, which you use to tell OpenEdge to convert all characters to uppercase during input. The case table you specify is used to establish case rules for the code page used in OpenEdge's memory. The memory code page is set with the Internal Code Page (`-cpinternal`) parameter. The default value is BASIC.
Conversion map (-convmap)	The conversion map file that OpenEdge uses for code page conversions, collation orders, and case conversions. By default, OpenEdge uses *`OpenEdge-install-dir`*`\convmap.cp` in Windows and *`OpenEdge-install-dir`*`/convmap.cp` on UNIX.

Property	Description
Server code page (-cpinternal)	The code page that OpenEdge uses in memory. For GUI clients, the `-cpinternal` code page should be the same code page the operating system uses. If you do not specify a value, OpenEdge uses the iso8859-1 code page by default.
Log code page (-cplog)	The code page that the database uses when it writes to a log file. By default, the database uses the internal code page. Choose a code page that agrees with your operating system and its devices. If you have a Unicode database, you might want to specify the log character set as Unicode so information is properly logged. If you do not specify a value, OpenEdge uses the iso8859-1 code page by default.

Advanced database configuration properties

Table 8: Advanced database configuration properties

Property	Description
Database service communication area size (-pica)	The value (which corresponds to the `-pica` command-line option) that sets the size of the database service communications area. This area is used for storage of after-image block write notifications intended for OpenEdge Replication. The minimum value is 4 and the maximum value is 8192; the value is entered in Kb units.
Nap time increments between access attempts of PICA queue. (-picanap)	The time in milliseconds before a process may access the PICA queue. The default value is 1.

Property	Description
Windows Event Level (-evtlevel)	The level of information written to the Windows Application Event Log. Choose from the following options: • **None** — No events are written to the Event Log. • **Brief** — Error and Warning messages are written to the Event Log. • **Normal** — Error and Warning messages are written to the Event Log along with any message that is normally written to the log file (`.lg`). This is the default value. • **Full** — Error, Warning, and Informational messages are written to the Event Log along with any messages generated by the Message Statement.
Lock table hash table size (-lkhash)	The size of the hash table that controls access to the lock table. **Note:** Progress Software Corporation recommends changing the value of this property only after contacting Technical Support.
Maximum number of database areas (-maxareas)	The highest area number available for use during the time the database is online.
Nap interval (-nap)	The minimum nap time in the spin lock algorithm. The default value is 10.
Nap time increment (-napinc)	The nap increment value. (Enterprise database only) **Note:** Do not edit this property unless directed to do so by Progress Software Technical Support.
Nap max interval (-napmax)	The maximum nap time in the spin lock algorithm before the nap time resets to the minimum. The default value is 250.
Nap time steps between nap time (-napstep)	The steps between the nap increment. (Enterprise database only) **Note:** Do not edit this property unless directed to do so by Progress Software Technical Support.

Property	Description
Page writer max buffers (-pwwmax)	The maximum number of modified database buffers to write to disk during a buffer scan cycle. The default value is 25.
Page writer queue delay (-pwqdelay)	The number of milliseconds between scans of the page writer queue. The default value is 100.
Page writer queue minimum (-pwqmin)	The minimum number of buffers required on the page writer queue before an APW writes them to disk. The default value is 1.
Page writer scan (-pwscan)	The number of buffers that an APW should scan during a buffer scan cycle. The default value is 1.
Page writer scan delay (-pwsdelay)	The number of seconds of delay between each scan of the database buffers by the APW. The default value is 1.
Pending connection timeout (-PendConnTimeout)	The amount of time a client has to connect to a server before the broker clears the client's reservation.
Semaphore sets (-semsets)	The number of semaphore sets available to the database broker. On single-processor systems, OpenEdge uses semaphores to synchronize the activities of server and self-service client processes that are connected to a database. By default, each database has an array of semaphores, one for each user or server. Each process uses its semaphore when it must wait for a shared resource. Semaphores are not used for single-user sessions or for client sessions connecting to a remote database on a server system. By default, this value is set to 1. When more than 1,000 users connect to a single database, there might be high contention for the semaphore set. If there is a lot of semaphore contention on a system, using multiple semaphore sets helps alleviate this contention and improves performance on high user counts.

Property	Description
Shared memory overflow size (-Mxs)	In kilobytes, the size of the shared-memory overflow area. This parameter replaces the default value of the shared-memory overflow area; it does not increase it. The overflow area is appended to the shared-memory area. The default value is 0.
Pin shared memory segments (-pinshm)	Prevents the database engine from swapping shared memory contents to disk.
Maximum shared-memory segment size (-shmsegsize)	The maximum number of bytes, specified in MB or GB, in a shared memory segment.
Spin lock retries (-spin)	The number of times a process tries to acquire a latch before pausing. If the process cannot acquire the resource's latch, it continues the attempt. This iterative process is called spinning. The default value is based on the number of CPUs of the system. The default is set to 10,000 for single-CPU systems. For multi-CPU systems, the default is equal to 6,000 * # of CPUs. If the value of this property is greater than zero, a spin lock algorithm is used for shared-memory data structures. When a process has to lock a shared-memory structure, the process attempts to acquire the latch for that structure up to the specified number of times. If the process has not acquired the latch in the specified number of attempts, then the process pauses, or naps. The length of the pause increases gradually if the process repeatedly fails to acquire a latch. After the allotted nap time, the process wakes up and attempts to acquire the latch again. If it fails to acquire the latch, it retries up to the number of attempts specified.
Storage object cache size (-omsize)	The size of the object cache for all database objects.
TXE commit lock skip limit (-TXESkipLimit)	Sets the maximum SHARE/UPDATE locks that can skip the queued COMMIT locks. **Note:** Do not edit this property unless directed to do so by Progress Software Technical Support.

Property	Description
User MUX latches (-mux)	A value that controls the granularity of access to large database resources in shared memory. **Note:** Do not edit this property unless directed to do so by Progress Software Technical Support.
SQL lock Wait timeout (-SQLLockWaitTimeout)	The number of seconds to wait when a lock conflict occurs. This value applies to all conflicts that occur while using SQL applications. The default value is 5 seconds.
User notify time(In Seconds) (-usernotifytime)	The time, in seconds, of how often a client polls for notification of a schema change. The default is 0.
STS Logging Level (-stslogginglevel)	The logging level value for STS client interface logging. The logging level: • **0** — none • **1** — fatal errors only • **2** — all errors • **3** — all errors and warnings • **4** — all errors, warnings, and informational messages • **5** — all errors, warnings, informational, and debug messages Values less than 0 generate an error; values greater than 5 are reset to 5 internally.
Change Data Capture cache size (-cdcsize)	The size, in kbytes, of both the primary and secondary CDC caches combined. The default value is 2000 if CDC is enabled, and 200 if CDC is not enabled.
Database notify time (-dbnotifytime)	The time interval, in seconds, of when a notification is sent about the synchronization betweeen server and clients database administration online operations. The default and minimum value is 10 seconds. The maximum value is 300 seconds.

Diagnostics database configuration properties

Table 9: Diagnostics database configuration properties

Property	Description
Diagnostic directory (-diagDir)	Specifies the directory where the subdirectories for each diagnostic event are created and the diagnostic tracking log file is written. This is a fully-qualified directory specification. The directory name can be a maximum of 255 bytes. If you do not specify a directory with `-diagDir`, the database directory is used. Though you can specify an absolute path or a relative path, the property validates only an absolute path. It skips the validation of relative path.
Diagnostic events value (-diagEvent)	Specifies the database events for collecting diagnostic data. This is a character string of comma-separated keywords specifying triggering events and levels.
Diagnostic event level (-diagEvtLevel)	Specifies the level of diagnostic data to collect. The minimum and maximum values (integer) are -15 and 15 respectively.
Diagnostic field separator (-diagFS)	Specifies the separator to use when writing diagnostic data to a CSV-formatted file. The available options are: **comma**, **tab**, **semi-colon**, and **pipe**. If you need any options other than the ones specified here, you can add them in `OtherArgs`.
Diagnostic data format (-diagFormat)	Specifies the format of the diagnostic data files. The available formats are **CSV** and **JSON**.
Diagnostic pause length (-diagPause)	Specifies the number of seconds to pause after recording data for a diagnostic event and before processing the event. The minimum and maximum values are 0 and 1920 (30 minutes) respectively.
Diagnostic prefix value (-diagPrefix)	Specifies a common prefix to standard diagnostic data files. This is a string of printable ASCII characters that can be used to prefix standard diagnostic data files. The prefix can be a maximum of 16 bytes. The fully-specified file name cannot exceed 300 bytes, which includes the specification of diagnostic directory, event sub-directory, and the data file. For example, *`prefix`*`locktable_summary.csv`.

Property	Description
Omit log messages (-omitLgMsgs)	Specifies the messages you do not want written to the database log file. You can specify up to 16 message numbers in a comma-separated list, with no embedded spaces in the list. The message omission list is on a per database basis, and only applies to messages being written to the database .lg file. Messages written to the screen are not suppressed. The values specified at startup are available via the _DbParams VST. The values can be read and updated.
Log file truncate time (-lgTruncateTime)	Specifies a time to truncate the log file in *HH:MM* format. You must specify both `-lgTruncateFrequency` and `-lgTruncateTime`. If you specify only one of the parameters, no truncation occurs.
Log file truncate frequency (-lgTruncateFrequency)	Specifies how often to truncate your log file. You must specify both `-lgTruncateFrequency` and `-lgTruncateTime`. If you specify only one of the parameters, no truncation occurs.
Log file truncate size (-lgTruncateSize)	Specifies a size, in MB, for truncating the log file. The default value is 0.
Log file archive enable (-lgArchiveEnable)	Specifies whether the database log file should be archived before it is truncated. If the archive file can not be created for any reason, the log file is not truncated. This ensures that no log file entries are lost.
Log file archive directory (-lgArchiveDir)	Specifies a directory for the archived log file. The maximum size of a fully-qualified directory specification is 255 characters.
Limit log file payload (-limitLgPayload)	Prevents message payload from being written to the database log file. Message payload is defined as the user name, tty name, IP address, and connection hostname in a message. Specify if payload is included or omitted. • **0** — Limit payload is off; the complete contents of messages is written to the database log file. • **1** — Limit payload is on; payload data within a message is not written to the database log file, and is replaced with the string, "<omitted>". All messages written to the database log file (dbname.lg), and messages that are written to both the log file and the screen, honor the -limitLgPayload setting. Messages only written to the screen, do not check the payload setting. Other log files, such as for SQL or Replication, honor the payload setting when the information written to the log file is obtained from the database connection information.

Statistics Collection Configuration database properties

Table 10: Statistics Collection Configuration database properties

Property	Description
Base index number (-baseindex)	The starting index number in the range of indexes for which you want to track access statistics. The default value is 1.
Index range size (-indexrangesize)	The number of indexes for which you want to track access statistics.
Base table number (-basetable)	The starting table number in the range of tables for which you want to track access statistics. The default value is 1.
Base user table number (-baseusertable)	The starting table number for the range of user tables for which you want gather information in the `_UserTableStat` VST. The default value is 1.
Base user index number (-baseuserindex)	The starting index number for the range of indexes for which you want gather information in the `_UserIndexStat` VST. The default value is 1.
Table range size (-tablerangesize)	The number of tables for which you want to track access statistics

SQL-92 Configuration database properties

Table 11: SQL-92 Configuration database properties

Property	Description
Java classpath (-classpath)	The pathname of the classpath
Number of open SQL cursors (-SQLCursors)	The number of open cursors per connection
SQL stack size (-SQLStack)	The size, in MB, of the SQL Stack
SQL statement cache size (-SQLStmtCache)	The number of statements allowed in the SQL statement cache
SQL sorting memory (-SQLTempBuff)	The size of the temporary table buffer in memory
SQL sorting on disk (-SQLTempDisk)	The size of the temporary table for backup storage

Property	Description
SQL temporary data page size (-SQLTempPgSize)	The size of the temporary table data page
Maximum number of JTA transactions (-maxxids)	The number of JTA transactions simultaneously allowed
SQL Truncate Too Large (-SQLTruncateTooLarge)	Truncates the data returned to the user when its size exceeds the column size.

TLS Configuration database properties

Table 12: TLS Configuration database properties

Property	Description
Enable SSL for remote connections (-ssl)	Specifies that all connections to this server socket must use TLS.
Key alias name (-keyalias)	Sets the alias name within the keystore of the private key and digital certificate entry to use to authenticate all connections to this server socket. If the key alias name is not specified, the server socket uses the default_server server certificate alias.
Key alias password (-keyaliaspasswd)	Sets the password to use for accessing the private key and digital certificate. You must specify a password when you specify the `-keyalias` option, and the password must be encrypted. You must specify the password as an encrypted value that you can obtain using the genpassword utility located in the `bin` directory of your OpenEdge installation. If you use the default_server server certificate, it also has a default password that you do not need to specify.
Disable SSL session cache (-nosessioncache)	Disables session caching. (Session caching allows a client to reuse a previously established session if it reconnects prior to the session cache timeout expiring.)
SSL server ciphers (-sslciphers)	Accepts the cryptographic cipher that is set for the client.
SSL server protocols (-sslprotocols)	Accepts the cryptographic protocol that is set for the client

Agent database configuration properties

Table 13: Agent database configuration property

Property	Description
Monitored	Starts the database monitoring agent automatically.This option is selected by default.

Configure the Environment variables property

Environment variables

To specify the environment variables for database execution, select the **Environment Variables** category instead of updating the same in the `$DLC\properties\conmgr.properties` file for OpenEdge release versions that are greater than 11.7.x Enter name-value pairs for environment variable settings. Any values you set here will override the prior values set for the same environment variables in the operating system.

Note: The characters `.`, `#` and `[]` are not allowed in the key-value pairs which are part of the environment variables

---

Edit the database default configuration

OpenEdge Management and OpenEdge Explorer provide a default database configuration. You can modify the existing configuration and then either save it as a replacement for the default configuration or create a new configuration.

Note: When you create a new configuration, the configuration automatically uses the pre-defined OpenEdge defaults.

To edit a database default configuration:

1. Click **Resources** from the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database whose default configuration you want to edit.
3. Click the Edit icon. The **Database** details page appears.
4. In the **Command and Control** section of the page, click **Configuration**. The **Database Configuration** page appears.
5. Click the **configuration.<*databasename*>.defaultconfiguration**. The **Database Configuration** page appears, with the default configuration properties organized into the categories described in View the various database configuration properties on page 23.
6. Click **Edit** and make the modifications.
7. Click **Save**.

Create a database configuration

OpenEdge Management and OpenEdge Explorer provide a predefined default database configuration; however, you can create a new configuration by modifying the default settings.

To create a database configuration:

1. Click **Resources** from the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database whose default settings you want to modify to create a new configuration.
3. Click the Edit icon. The **Database** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Database Configuration** page appears.
5. Click **configuration.<*databasename*>.defaultconfiguration**. The **Database Configuration** page appears and displays the default configuration properties.
6. Click **Create**.
7. Type a new configuration group name in the field provided, and click **Save**. The **Database Configuration** page appears.
8. Click **Edit** to set the properties of the new database configuration. (Refer to for a description of each property.)
9. When you finish, click **Save**.

To make this new configuration the default, you must select it as the default on the **Database Configuration** page.

To assign the new configuration to a database, see Assign a different default database configuration on page 39.

Assign a different default database configuration

OpenEdge Management and OpenEdge Explorer provide a default database configuration. You can modify that default configuration, or you can create one or more additional configurations and assign one of them as the default.

To assign a different default database configuration:

1. Click **Resources** from the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database to which you want to assign a different default configuration.
3. Click the Edit icon. The **Database** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Database Configuration** page appears.
5. Click **Edit**.
6. From the **Default configuration** dropdown, select the default configuration you want to use.
7. Click **Save**.

View the server group

The server group specifies the logical collection of server processes to start.

You can apply properties to a database server group by either using the default configuration or creating and then specifying a configuration of your own choosing.

You can set or edit any of the database server group properties from OpenEdge Management or OpenEdge Explorer. The properties are organized into the **General** and **Advanced** categories.

To view the server group:

1. Click **Resources** from the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database whose server group you want to view.
3. Click the Edit icon. The **Database** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Database Configuration** page appears.
5. Click **servergroup.<*databasename*>.defaultconfiguration.defaultservergroup.** The **Database Configuration** page for the server group appears.

From the **Database Configuration** page, you can do the following:

- View a read-only display of the **General** and **Advanced** database server group properties as described in Database server group properties on page 40.
- Click **Edit** to modify the database server group properties. For details, see Edit a database server group on page 43.
- Click **Create** to create or modify a server group. For details, see Create a database server group on page 43.
- Click **Delete** to delete a server group, provided it is not running. For details, see Delete a database server group on page 48.

Database server group properties

The following topics describe the database server group properties.

General database server group properties

Table 14: General database server group properties

Property	Description
Service name or port	The service name or port number through which clients may connect to the database. The default value is 0. **Note:** The service name or port number must represent a non-zero value in order to start a networked database broker. When you define multiple server groups for the same configuration, you must assign a unique service name or port number to each server group.
Number of servers (-Mpb)	The maximum number of server processes that can be started to service clients. The default value is 0.
Message buffer size (-Mm)	The standard message buffer size in bytes. OpenEdge uses message buffers to move records (messages) between servers and remote clients. Records (plus 40-byte headers) larger than the message buffer size are fragmented into multiple messages. If your database records are large, increase this parameter to avoid record fragmentation. However, if the network works more efficiently with small messages, reduce the **Message Buffer size** value and fragment larger records. The default value is set dynamically based on your platform (UNIX or Windows) and the type of database (Personal, Workgroup, or Enterprise).
Reporting interval (-rptint)	The interval, in minutes, at which the server polls for license usage statistics that are reported in the usage report file. The default value is 1.

Property	Description
TCP/IP Version (-ipver)	The TCP/IP version—IPv4 or IPv6—that the database broker uses to open the database port. The default value is IPv4.
Client Type (-ServerType)	One of the following client types: • **4GL only** — The server group can service requests only from ABL clients. The settings are the same as those for the **Both 4GL and SQL** option. • **SQL only** — The server group can service requests only from SQL clients. The settings are the same as those for the **Both 4GL and SQL** option, except that the **Message buffer size** and **Reporting interval** settings are unavailable. • **Both 4GL and SQL** — The server group can service requests from both ABL and SQL clients. **Note:** Remember that the client type choice you make here is honored only when you have installed the proper licenses for that client. For example, if you selected the **4GL only** option during the OpenEdge installation, choosing the **SQL only** option or the **Both 4GL and SQL** option in the management console has no effect. However, if you chose the **Both 4GL and SQL** option during the OpenEdge installation, choosing **SQL only** in the management console is valid.

Advanced database server group properties

Table 15: Advanced database server group properties

Property	Description
Maximum clients per server (-Ma)	The maximum number of remote users per database server. The default value is 0.
Minimum clients per server (-Mi)	The minimum number of remote users on a server before the broker starts another server (up to the maximum number of servers). The default value is 1.
Maximum dynamic port (-maxport)	The port number that is the highest in a specified range of server port numbers accessible to a client. You specify the lowest port number with the **Minimum dynamic port** property. The default value is set dynamically based on your platform (UNIX or Windows) and the type of database (Personal, Workgroup, or Enterprise).

Property	Description
Minimum dynamic port (-minport)	The port number that is the lowest in a specified range of server port numbers accessible to a client. You specify the higher port number with the **Maximum dynamic port** property. Ports below 1025 are usually reserved for system services. The default value is set dynamically based on your platform (Unix or Windows) and the type of database (Personal, Workgroup, or Enterprise).
Host (-H)	The database's host machine.

Edit a database server group

For a database instance you added to OpenEdge Management or OpenEdge Explorer, you can edit any of the database server group properties.

To edit a database server group:

1. Click **Resources** from the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database whose server group you want to edit.
3. Click the Edit icon. The **Database** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Database Configuration** page appears.
5. Click the **servergroup.<*databasename*>.defaultconfiguration.defaultservergroup** you want to edit. The **Database Configuration** page for the server group appears.
6. Click **Edit** and make the modifications. (For details about the properties in the **General** and **Advanced** categories, refer to .)
7. Click **Save**.

Create a database server group

You can create a new database server group from OpenEdge Management or OpenEdge Explorer.

To create a database server group:

1. Click **Resources** from the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database where you want to create a new server group.
3. Click the Edit icon. The **Database** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Database Configuration** page appears.
5. Click the **servergroup.<*databasename*>.defaultconfiguration.defaultservergroup** you want to edit. The **Database Configuration** page for the server group appears.

6. Click **Create**.
7. Type a new server group name in the field provided, and click **Save**. The **Database Configuration** page appears.

   You can click **Edit** to set the properties of the new database server group. (For details about the properties in each category, refer to .) When you finish, click **Save**.

Start or stop the database

OpenEdge Management and OpenEdge Explorer support starting and stopping both local databases and databases running under a remote AdminServer on another machine. A local database resides on the same machine as OpenEdge Management or OpenEdge Explorer.

Before you can start or stop a database, you must first connect to the AdminServer that contains a definition of the database.

To start or stop a database, refer to the *Starting or Stopping OpenEdge resources* topic in *OpenEdge Management: Servers, DataServers, Messengers, and Adapters*. The database agent works only if the database is running. Therefore, when you stop a database, the agent status automatically changes to **Not Running**. However, when you start a local database whose **AgentMonitored** property is disabled, the agent does not automatically start. You must specifically click **Start Agent** in the Database control page.

Note: You can run only one configuration for a database at a time. You can set the database configuration to start automatically when the AdminServer starts. For details, see Start a database automatically on page 44.

Start a database automatically

You can set a database to start automatically when the AdminServer starts.

To start a database automatically:

1. Click **Resources** from the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database you want to start automatically.
3. Click the Edit icon. The **Database** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Database Configuration** page appears.
5. Click **Edit**.
6. Select the **Autostart** option by clicking in the check box.
7. Click **Save**.

Background writers and watchdog processes

On shared-memory systems, setting database background writer and watchdog processes can improve database performance. Background writers continuously perform overhead functions in the background.

Background writers

There are three types of background writers:

- **Asynchronous Page Writer (APW)** — The APW improves performance by performing overhead operations in the background. These operations provide available buffers, reduce the number of buffers that OpenEdge reads before writing to disk, and reduce the overhead associated with before-image checkpoints (the process of synchronizing the buffer pool of modified blocks to the database). You can start and stop an APW at any time without shutting down the database.
- **Before-image Writer (BIW)** — The BIW improves performance by continually writing before-image buffers to disk. These writes occur in the background. BIWs are optional; however, they are highly recommended for improving I/O performance. You can run only one BIW per database. You can start and stop a BIW at any time without shutting down the database.
- **After-image Writer (AIW)** — The AIW improves performance by continually writing after-image (AI) buffers to disk soon after OpenEdge fills the buffers. The after-imaging feature lets you recover a database that was damaged when a failure caused the loss of the database or primary recovery (BI) area. When you enable after-imaging, OpenEdge writes notes to the after-image files that contain a description of all changes to the database. You can run only one AIW process per database. You can start and stop an AIW at any time without shutting down the database.

The watchdog process

The watchdog process improves performance by cleaning up after improperly terminated processes. It releases locks, backs out any live transactions, and releases shared-memory locks. It also disconnects and cleans up the server's remote clients.

View database status

You can view the status of a database's configurations, server groups, and auxiliary processes through OpenEdge Management or OpenEdge Explorer. You can also view the status of the database monitoring agent.

Note: Most of the broker instances provide a status view, enabling you to monitor the status of multiple brokers simultaneously. You can monitor instances located on the same or different servers.

To view the status of a database:

1. Click **Resources** from the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database whose status you want to view.
3. Click the Edit icon. The **Database** details page appears.
4. In the **Command and control** section of the page, click **Control**. The **Database Control** page appears.
5. Review the following status information for the database:
   - **Database information** — The database name, the active port or ports, and the name of the configuration being used by the database.
   - **Database status** — The status for the database and monitoring agent.

- **Database auxiliary status** — The status of the After image page writer, the Before image page writer, the Asynchronous page writers, and the Watchdog process for this server group.
- **Properties** — The status of the **Monitor** and **Enabled** settings: enabled (checked) or disabled.

View database process details

You can access real-time details and statistics that provide you with snapshot information about a running database at the point you access this information from the **Control** page. You can review this information to help you assess the database's performance.

To access processing details:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database whose status you want to view.
3. Click the Edit icon. The **Database** details page appears.
4. In the **Command and control** section of the page, click **Control**. The **Database Control** page appears.
5. Click a **PID** link. The following sections appear:
   - The **Process summary** section identifies the **Process name** and **Process start time**. **User id** and **Group id** values appear when UNIX-based data is shown. The **Parent pid** provides the identifier number associated with the process that spawned this current process.
   - The **Process statistics** section presents details about the database's real-time operational status. Values presented without parentheses identify that the processing time determined since the last scheduled polling interval, as noted, has occurred. Values presented within parentheses have been calculated based on information obtained since the start of the process.

Process statistics data

The following table identifies and describes the fields of information presented in the **Process statistics** section.

Table 16: Process statistics data

Field	Description
Resident size	The physical size of the process as defined by the host system.
Virtual size	The virtual size of the process as defined by the host system.
CPU	The percentage of time spent using the CPU in either the user or kernel mode since the last scheduled poll.
Weighted CPU	The percentage of time spent using the CPU in either the user or kernel mode since the last scheduled poll divided by the number of CPU processors on the system. This value appears only when there is more than one CPU process on the system where the process is running.
User time	The amount of CPU time spent in the user mode since the last scheduled poll.

Field	Description
Kernel time	The amount of CPU time spent in the kernel mode since the last scheduled poll.
Process time	The sum of the values that appear in the **User time** and **Kernel time** fields.

Delete a database, database configuration, or database server group

You can delete a database, provided it is not running, and you can delete a database configuration or database server group, provided it is not currently in use by a database.

Delete a database

When you no longer need a particular database, you can delete the database and all associated configuration information from the management console. When you delete a database from the console, you are not deleting the actual, physical database files; you are only deleting the database definition from the `conmgr.properties` file. After you delete the database, it no longer appears in the console.

You cannot delete a database that is running. You must first stop the database and then delete it.

To delete a database from the management console:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database whose status you want to delete.
3. Click the Edit icon. The **Database** details page appears.
4. Click **Delete** and then click **OK** to confirm the deletion.

   The **OpenEdge Management Resources** page appears. A set of next-step options that are related to resources are available for your use.

Delete a database configuration

You can delete a database configuration from the management console.

If a configuration is running, you cannot delete it. You must first stop the configuration, and then delete it.

To delete a database configuration from the management console:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database you want to delete.
3. Click the Edit icon. The **Database** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Database Configuration** page appears.

5. In the **Configuration and Server Group Links** section of the page, click the database configuration you want to delete. The configuration information appears.
6. Click **Delete**, and then click **OK** to confirm the deletion.

Delete a database server group

You can delete a database server group from the management console.

If the configuration associated with the server group is in use, you cannot delete it. You must first stop the configuration, and then delete the server group.

To delete a database server group:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database whose server group you want to delete.
3. Click the Edit icon. The **Database** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Database Configuration** page appears.
5. In the **Configuration and Server Group Links** section of the page, click the server group you want to delete. The server group information appears.
6. Click **Delete**, and then click **OK** to confirm the deletion.

Configure an OpenEdge Replication-enabled database

To configure your source database and target database for OpenEdge Replication, you must configure the replication property files for both the OpenEdge Replication source database and the OpenEdge Replication target database.

In the *OpenEdge-install-dir*`/properties` folder, OpenEdge Replication provides two sample property files: `source.repl.properties` and `target.repl.properties`. You can save these files and modify them, using either OpenEdge Management or OpenEdge Explorer, or a text editor, so that you can use them with your source and target OpenEdge Replication databases. Because `source.repl.properties` and `target.repl.properties` are samples, some of the properties and values are generic; it is up to you to customize the properties to reflect the settings you want to use with your own source and target databases.

When you use OpenEdge Management or OpenEdge Explorer to make configuration changes to the Replication server or Replication target properties, the changes are reflected in the applicable property file.

Rules for setting properties for a Replication-enabled database

From the management console, you can configure various properties for a database enabled for Replication. The options you have in the console for configuring these Replication-related properties depend upon the following rules:

- For a database **notenabled** for Replication:
  - If the Replication property file **does not** exist, no Replication links are available.
  - If the Replication property file **does** exist but is empty, a **Server (create new)** link and an **Agent (create new)** link appear.

Note: You can configure transition only when one of these groups exists.

- If the Replication property file **does** exist and **is not** empty, then all groups defined in the property file appear.

- For a database **enabled** for Replication:
  - If the Replication property file **does not** exist, Replication-related links still appear. However, you can create only the group for which the database is enabled. In other words, you can create the Server group for source-enabled databases and the Agent group for target-enabled databases.
  - If the Replication property file **does** exist, only the groups that are applicable to the database enablement appear. For source enablement, **Server**, **ctrl-agents**, and **transition** appear. For target enablement, **Agent** and **transition** appear.
  - If the Transition group database-role is set to **reverse**, you either see the existing group for the reverse role or are allowed to create a new group for the reverse role.
  - If you specify **Alternate buffer pool** blocks for a Replication target database, the property setting is ignored.

Set Replication-related properties

If you are familiar with OpenEdge Replication and its configuration, see the following topics for details about the properties you can configure through the management console:

- Set OpenEdge Replication Agent properties on page 49
- Set OpenEdge Replication Control Agent properties on page 52
- Set OpenEdge Replication Server properties on page 50
- Set OpenEdge Replication Transition properties on page 53

If you are not familiar with OpenEdge Replication and its configuration, see *OpenEdge Replication: User Guide* for details about setting the properties.

Set OpenEdge Replication Agent properties

You set OpenEdge Replication Agent properties to define the configuration for the local agent running for the target database.

Note: If you are not familiar with OpenEdge Replication and its configuration, see *OpenEdge Replication: User Guide*.

To set the OpenEdge Replication Agent properties:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database whose Replication Agent properties you want to set.
3. Click the Edit icon. The **Database** details page appears.

4. In the **Command and control** section of the page, click **Configuration**. The **Database Configuration** page appears.
5. In the **OpenEdge Replication Group Links** section, click **Agent**. The **Database Replication Configuration** page opens.
6. Set the following properties:

   - **Agent name** — Specifies the OpenEdge Replication agent name. By convention, the agent name should match the name specified in the source database properties file.

     ALL is not allowed as an agent name. Each agent must have a unique name.

   - **Database** — Specifies the source database name.
   - **Connection Timeout** — Specifies how many seconds the OpenEdge Replication agent will wait for connection from the OpenEdge Replication server before the Replication agent shuts itself down.

     The seconds value must be greater than or equal to 120, and less than or equal to 86,400.

     Using this property means you do not have to do a forced shutdown on your target database. If the OpenEdge Replication agent does not receive a connection attempt from the OpenEdge Replication server before the number of seconds specified have elapsed, the OpenEdge Replication agent will terminate and allow some limited system-level target database connections.

   - **Polling delay minimum** — Specifies the minimum value, in milliseconds, for a polling delay.

     By default, the polling delay starts at 5 milliseconds and automatically increases during periods of inactivity to a maximum of 500 milliseconds.

     The milliseconds value must be greater than or equal to one, and less than or equal to ten.

   - **Polling delay maximum** — Specifies the maximum value, in milliseconds, for a polling delay.

     By default, the polling delay starts at 5 milliseconds and automatically increases during periods of inactivity to a maximum of 500 milliseconds.

     The milliseconds value must be greater than or equal to 500 and less than or equal to 1000.

   - **Agent minimum port** — Specifies the minimum TCP port number. The agent selects a port in a range between the values specified by the minimum port and the maximum port.

     The minimum port number value must be greater than 1024 and less than the value of the maximum port.

   - **Agent maximum port** — Specifies the maximum TCP port number for the agent.

     The maximum port number value must be greater than the minimum port + 1, and less than the maximum allowable port number on the system.

     For UNIX the maximum port number is 65534. For Windows the maximum port number is 32765.

   - **Other arguments** — Any other arguments for which there is no defined property available.

7. Click **Save**.

Set OpenEdge Replication Server properties

You set the OpenEdge Replication Server properties to specify server and agent property values.

Note: If you are not familiar with OpenEdge Replication and its configuration, see *OpenEdge Replication: User Guide***before** setting the properties.

The server properties are divided into the following categories:

- General
- Advanced

To set the OpenEdge Replication Server properties:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database whose Replication Server properties you want to set.
3. Click the Edit icon. The **Database** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Database Configuration** page appears.
5. In the **OpenEdge Replication Group Links** section, click **Server**. The **Database Replication Configuration** page opens.
6. Set the following **General** properties:

   - **Database** — Specifies the source database name.
   - **Defer agent startup** — Specifies for how long, in minutes, the server attempts to connect to an agent if the first connection attempt is unsuccessful. The minutes value must be greater than or equal to zero and less than or equal to 10080.
   - **Keep alive** — Specifies a timeout period for communications between a server and its agents. If a connection between the server and the agent is not verified before the timeout expires, failure recovery begins.

     By default, this property is enabled and has a value of 300 seconds. The minimum value is 60 seconds; there is no maximum value.
   - **Server transition timeout** — Specifies the number of seconds the target database waits before it performs auto-transition. This property is ignored when transition is set to manual.

     The value is incremented by the sum of the connect-timeout for all configured agents. The minimum and maximum values are 0 and 86400 seconds respectively.
   - **Server transition** — Specifies how to transition the target database to a normal database.

     If a synchronous agent or an asynchronous critical agent cannot reconnect with the source database within the time specified in the transition-timeout property, the following occurs:

     - **auto** — The agent automatically transitions the target database.
     - **manual** — The agent listens for the server until it reconnects with the source database or the DSRUTIL command is executed. The `DSRUTIL db-name -C` transition command transitions the target database.

7. Set the following **Advanced** properties:

   - **Other arguments** — Any other arguments for which there is no defined property available.
   - **Schema lock action** — Specifies the action an agent takes if an exclusive schema lock is not granted. Possible actions are:

     - **wait** — The agent waits until the exclusive schema lock is granted. The server blocks until the exclusive schema lock is granted.
     - **force** — The agent attempts to acquire the exclusive schema lock five times. If the fifth attempt fails, the agent disconnects all users from the target and makes another attempt. If the last attempt fails, the server and all agents terminate. When schema update activity completes, the server and target can be restarted.

- **Polling delay minimum** — Specifies the minimum value, in milliseconds, for a polling delay. By default, the polling delay starts at 5 milliseconds and automatically increases during periods of inactivity to a maximum of 500 milliseconds.

  The value must be greater than or equal to one and less than or equal to ten.
- **Polling delay maximum** — Specifies the maximum value, in milliseconds, for a polling delay. By default, the polling delay starts at 5 milliseconds and automatically increases during periods of inactivity to a maximum of 500 milliseconds.

  The value must be greater than 500 and less than 1000.
- **Server port minimum** — The minimum port available.
- **Server port maximum** — The maximum port available.
- **Agent shutdown action** — Specifies the action an agent takes during a shutdown when the Replication server ends. Possible actions are:
  - **recovery** — The agent remains active but in a standby state waiting for the replication server to reconnect.
  - **normal** — The agent terminates; the target database stays up.

8. Click **Save**.

Set OpenEdge Replication Control Agent properties

You set the OpenEdge Replication control agent properties to define for the server which agents it will contact, where it will contact them, and how the agent should perform. Each server group supports up to two control agents.

Note: If you are not familiar with OpenEdge Replication and its configuration, see *OpenEdge Replication: User Guide***before** setting the properties.

To set the OpenEdge Replication Control Agent properties:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database whose Replication Control Agent properties you want to set.
3. Click the Edit icon. The **Database** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Database Configuration** page appears.
5. In the **OpenEdge Replication Group Links** section, click **Control-Agent**. The **Database Replication Configuration** page opens.
6. Click **Edit**.
7. Set the properties as follows:
   - **Database** — Specifies the source database name.
   - **Host** — Specifies to the server which host the agent will start on.
   - **Port** — Specifies which port the server uses to connect to the target database. The port number specified must be the same port specified with the `-S` parameter when the target database broker was started.
   - **Connection Timeout** — Specifies how many seconds the OpenEdge Replication agent will wait for a connection from the OpenEdge Replication server before the Replication Agent shuts itself down.

The seconds value must be greater than or equal to 120, and less than or equal to 86,400.

Using this property means you do not have to do a forced shutdown on your target database. If the OpenEdge Replication agent does not receive a connection attempt from the OpenEdge Replication server before the number of seconds specified elapses, the OpenEdge Replication agent will terminate and allow some limited system-level target database connections.

This property is also used by the server when reconnecting to the agent after communication has been lost.

- **Maximum message** — The maximum number of bytes used for the TCP/IP communication messages.
- **Replication method** — Asynchronous or synchronous.
- **Critical agent** — Specifies whether the agent is critical. (Select the check box for a critical agent.)

  A critical agent is an asynchronous agent for the target database that can become the source database if the source database becomes unavailable.
- **Other arguments** — Any other arguments for which there is no defined property available.

8. Click **Save**.

Set OpenEdge Replication Transition properties

You set the Transition properties to specify the transition process values, which are used when the target database becomes a normal database after a failure.

Note: If you are not familiar with OpenEdge Replication and its configuration, see *OpenEdge Replication: User Guide***before** setting the properties.

The transition properties are divided into the following categories:

- General
- Start Up
- After Image
- Back Up

To set the OpenEdge Replication Transition properties:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the database whose Replication Transition properties you want to set.
3. Click the Edit icon. The **Database** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Database Configuration** page appears.
5. In the **OpenEdge Replication Group Links** section, click **Transition**. The **Database Replication Configuration** page opens.
6. Set the following **General** properties:
   - **Database role**:
     - **reverse** — The role of the database is reversed: A source database becomes a target database, and a target database becomes a source database.

- **normal** — The role of the database becomes that of a normal database; the database is no longer enabled for replication once the transition is performed. This is the default value.

- **Responsibility**:
  - **primary** — This is the primary database.
  - **secondary** — This is the secondary database.
- **Transition to agents** — Transitions to the first agent in the list when a failure occurs. If the first agent is not available, transition to the second agent in the list.

  You can specify any configured agent name as a valid value for this property. If you are specifying more than one value, separate the names by a comma. For example, `agent1,agent2`.

7. Set the following **Start Up** properties:

- **Restart after transition** — The database can be automatically restarted after transition is performed. When you select this option, the `*-startup-arguments` properties must be supplied, or the database startup will fail:
  - If the database role is normal, you must specify the `normal-startup-arguments`.
  - If the database role is reversed, you must specify the `source-startup-arguments` and the `target-startup-arguments`.
- **Source startup arguments** — If the database is transitioned to a source database, these arguments are used when the database is started. The arguments are appended to the PROSERVE command (used to start the database).

  In most cases, the only argument specified here should be `-pf` followed by a parameter file name; for example, `-pf` *`db-name`*`.source.pf`.
- **Target startup arguments** — If the database is transitioned to a target database, these arguments are used when the database is started. The arguments will be appended to the PROSERVE command.

  In most cases, the only argument specified here should be `-pf` followed by a parameter file name; for example, `-pf db-name.target.pf`.

  Because the database is a target, you must also specify the `-S` *`port-number`* or *`service-name`* argument to listen on the TCP/IP port specified with `-S port-number` or `service-name`.
- **Normal startup arguments** — If the database is transitioned to a normal database, these arguments are used when the database is started. The arguments will be appended to the PROSERVE command.

  In most cases, the only argument specified here should be `-pf` followed by a parameter file name; for example, `-pf` *`db-name`*`.normal.pf`.

8. Set the following **After image** properties:

- **Begin after imaging after transition** — Select the option to automatically begin AI after a target-to-source database transition. After-imaging can be started for a database that has AI areas.
- **Automatically add after imaging extents** — This property instructs the transition process to automatically add AI areas to the database if:
  - It is transitioned to a source database.
  - There are currently no AI areas for the database.

If you do not select the option, AI areas are not added to the database. If you do select the option, AI areas are automatically added to the database by using the structure file specified in the `ai-structure-file` property.

- **Structure file containing after image area definitions** — The name of the structure file that contains the list of AI areas to add.

9. Set the following **Back Up** properties:

   - **Backup method** — Specifies the backup method performed before AI is enabled. The following values are valid:

     - **mark** — Marks the database backed up by using the `rfutil db-name -C mark backedup` command. Marking the database as backed up does not allow future AI extents to be used when recovering from a disaster

     - **full off-line** — Backs up the database offline by using the OpenEdge PROBKUP utility. The backup is performed in two steps. The first backup is a full backup, which is performed before AI is enabled for the database. The second backup is an incremental backup, which is performed after AI is enabled, and after the role of the database is changed.

     - **full on-line** — This backup is performed after the database is restarted, which occurs after the database has been transitioned.

   - **Full on-line and off-line backup arguments** — The arguments required for the full online and offline backups that are performed for the database. At a minimum, the target file or device must be specified in these arguments for both online and offline backups.

     To avoid overwriting a backup, do not use the same target file or device for both the backup and the incremental backup.

     Do not use backup validation parameters (such as `-vp` and `-vf`).

     The arguments should begin with `device-name`.

   - **Incremental backup arguments** — The arguments required for the offline incremental backup performed after AI is enabled and the database's role is reversed.

     The arguments should begin with `device-name`.

3

Configure DataServers

You can configure DataServer property settings and add new DataServer instances in OpenEdge Management and OpenEdge Explorer. You can also start, stop, and delete DataServer instances, as well as view their status and log files.

For details, see the following topics:

- Oracle DataServer configuration and administration
- MS SQL Server DataServer configuration and administration

Oracle DataServer configuration and administration

The OpenEdge® DataServer for Oracle allows the OpenEdge Application Development Environment (ADE) and applications created with OpenEdge in Windows to access certain Oracle databases.

The DataServer for Oracle is also supported on several UNIX and LINUX platforms. For details, see Progress Communities.

You can use OpenEdge Management or OpenEdge Explorer to establish and administer the configuration of a Progress OpenEdge DataServer for Oracle. The DataServer for Oracle installation provides one predefined DataServer broker (orabroker1). You can use this predefined component as a starting point for creating and configuring additional DataServer brokers. Each broker is referred to as an instance.

Use the Oracle DataServer broker

To work with the Oracle DataServer, double-click the **Oracle DataServer** folder from the management console list frame. There is one pre-defined default Oracle DataServer (orabroker1).

You can perform the following Oracle DataServer actions:

- Configure an Oracle DataServer broker on page 58
- Edit an Oracle DataServer broker configuration on page 65
- Create an Oracle DataServer broker on page 65
- Start or stop an Oracle DataServer broker on page 66
- Start an Oracle DataServer broker automatically on page 66
- View the status of an Oracle DataServer broker on page 66
- Delete an Oracle DataServer broker on page 65
- View the Oracle DataServer broker log files on page 68

Configure an Oracle DataServer broker

You can view or modify the following default configuration properties of an Oracle DataServer broker from OpenEdge Management or OpenEdge Explorer:

- Broker
- Server
- Environment variables

You can also create a new Oracle DataServer broker, view a broker's status, delete a broker you no longer need, or view the broker's log files.

Each of these tasks is described in the following sections.

View or modify Oracle DataServer broker properties

You can view or modify Oracle DataServer broker properties in the following categories: general properties, owner information, controlling NameServer information, Data Service, setting logging properties, and advanced features.

To view or modify broker properties for an Oracle broker:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the Oracle DataServer broker whose properties you want to view or modify.
3. Click the Oracle DataServer broker that you want to view. The **Oracle DataServer** details page appears.
4. In the **Command and control** section of the page, Click **Configuration**. The **Oracle DataServer Configuration** page appears. From this page, you can:
   - View a read-only display of the Oracle broker properties as described in Oracle DataServer broker properties on page 59.

- Click **Edit** to modify the broker properties. For details, see Edit an Oracle DataServer broker configuration on page 65.

Oracle DataServer broker properties

The following topics describe the Oracle DataServer broker properties.

Oracle DataServer broker General properties

Table 17: Oracle DataServer broker General properties

Property	Description
Working directory	Your general working directory. The default value is `@{WorkPath}`.
Operating mode	For DataServer brokers, the operating mode is state-aware. The default value is state-aware.
Port number	The type of the TCP/IP listening port that the DataServer broker uses. The default value for webspeed is 3055 and the default value for appserver is 3090.
Auto start	Starts the DataServer automatically when the AdminServer starts. The **Auto start** check box is cleared by default.

Oracle DataServer broker Owner Information properties

Table 18: Oracle DataServer broker Owner Information properties

Property	Description
Group name	(UNIX only) The name of the group.
Username	Your user name. The default is blank.
Password	Your password. The default is blank.

Oracle DataServer broker Controlling NameServer properties

Table 19: Oracle DataServer broker Controlling NameServer properties

Property	Description
Register with NameServer	A check box that indicates whether to register the DataServer with a controlling NameServer.

Property	Description
Controlling NameServer	Identifies the NameServer that the DataServer uses to find a broker that supports the correct application services. (If you did not select to register the DataServer with a NameServer, the field is dimmed and unavailable.)
Registration mode	How the broker specifies its hostname when registering with its controlling NameServer. This hostname information is passed onto a client application when it attempts to connect to a Data Service that the broker supports. The choices are: • **Register IP** — (Default) This setting is the most efficient mechanism, and can be used in most cases. It registers with the IP address of the machine where the broker is located. • **Register-LocalHost** — The broker registers with the hostname of the machine that it runs on. Use this setting when the broker runs on a machine with a single hostname and more than one IP address. • **Register-HostName** — The broker registers with the values specified in the hostName property. Use this setting when your clients need a fully qualified host name to connect to a broker in a different DNS domain.
Registration host name	Identifies the hostname value to be used if the **Registration mode** property is set to **Register-HostName**.

Oracle DataServer broker Data Service properties

Table 20: Oracle DataServer broker Data Service properties

Property	Description
Data service name list	Data services are services that are provided by a selected broker's servers. The default service name is the broker name. The default value is blank.
Supports default service	A check box that indicates whether the Oracle broker supports the default service. Select the check box to have the broker support the default service, or click to clear the check box and cancel support for the default service. The **Supports default service** check box is cleared by default.

Oracle DataServer broker Logging Setting properties

Table 21: Oracle DataServer broker Logging Setting properties

Property	Description
Broker log filename	The general broker log filename. The default is `@{WorkPath}\orabroker1.broker.log`.
Broker logging level	The amount of information written to the broker log. The possible values are as follows: • **None** — No log file is written. • **Error Only** — Only error-related information is written to the log file, which reduces the size of the log file. • **Basic** — Only some information is written to the log file. • **Verbose** — All broker activity is written to the log file. This might produce large files that use up disk space. • **Extended** — Information in addition to **Verbose**.
Append to broker log file	Create a new broker log file each time a broker is started, even if the broker log file specified already exists. The **Append to broker log** file check box is selected by default.
Broker logging entry types	A single entry or comma delimited list of logging entry types.
Broker log file threshold size	Specifies a limit on how big the log file can get (in bytes). When the log is full, the broker creates a new log file with a sequence number.
Maximum number of broker log files	Limit on how many broker log files will be kept on the system when there is a threshold limit. It represents the number of log files including the current on and includes any rolled over log files that already exist when the process started.

Oracle DataServer broker Advanced Features properties

Table 22: Oracle Advanced Features properties

Property	Description
Maximum client instances	The maximum number of client connections that the broker can support concurrently. The default value of 512 that is set for this parameter is high enough to ensure that the number of client connections is unlimited. Realistically, however, the system-level resources needed to support an unlimited number of clients might be exhausted before this limit is reached. If you see broker log entries that contain Exception Messages for "out of space" or "OutOfMemory" you might need to reduce the maximum number of client connections to a more reasonable value. Note, however, that lowering this value can cause some client requests to be rejected ("Exceeded Max Clients"). Configuring and starting multiple brokers to handle higher client loads alleviates the problem.
Priority weight (0-100)(dynamic property)	An integer value between 0 and 100 that influences the share of the workload that the selected broker receives. The larger the value, the heavier the load that is distributed to the broker. Using the **Priority weight** value, the NameServer distributes client requests to the same DataServer service across all DataServer brokers. The default value is 0. You can update this property dynamically. Any changes will affect all current and new brokers.
Registration retry (in seconds)	After registering with its controlling NameServer, the DataServer broker periodically sends "keep-alive" messages to the NameServer to let the NameServer know that the broker is still active. The **Registration retry** value is the number of seconds that pass between "keep-alive" messages. The default value is 30.

View or modify Oracle DataServer server properties

You can view or modify Oracle DataServer server properties in the following categories: **General** and **Logging Setting**.

To view or modify server properties for an Oracle broker:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the Oracle DataServer broker whose server properties you want to view or modify.
3. Click the Oracle DataServer broker that you want to view. The **Oracle DataServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Oracle DataServer Configuration** page appears. From this page, you can do the following:

- View a read-only display of the Oracle server properties as described in Oracle DataServer server properties on page 63\.
- Click **Edit** to modify the server properties. For details, see Edit an Oracle DataServer broker configuration on page 65.

Oracle DataServer server properties

The following topics describe the Oracle DataServer server properties.

Oracle DataServer server General properties

Table 23: Oracle DataServer server General properties

Server properties	Description

Server executable file (dynamic property)	The pathname of the DataServer executable that the broker runs. The defaults for the DataServer for Oracle on the Windows and UNIX platforms are as follows:
	• Windows:
	`@{Startup\DLC}\bin\_orasrv.exe`
	• UNIX:
	`OpenEdge-install-dir/_orasrv`
	You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.

Server startup parameters (dynamic property)	The startup parameters for the server. You should not modify the default parameters, which are: `-svub -S -X -N TCP -U X -P x -hs 0 -s 40` You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.
PROPATH	A list of directories OpenEdge searches to find procedures.

Server properties	Description
port number (dynamic property)	The minimum port number that you want to make available for each DataServer broker. The default is blank. You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.
Maximum port number (dynamic property)	The maximum port number that you want to make available for each DataServer broker. The default is blank.You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.

Oracle DataServer server Logging Setting property

Table 24: Oracle DataServer server Logging Setting properties

Server property	Description
Server log filename	The DataServer log filename. Enter a valid path name. Installed Default: `@{WorkPath}\orabroker1.server.log`

Set Oracle DataServer broker environment variables

You can set environment variables for the DataServer broker for Oracle. You can set one or more environment variables that:

- Are accessible from your ABL application
- Affect the environment in which your application runs

For the Oracle DataServer, there are no required environment variables in Windows.

To set Oracle DataServer broker environment variables:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the Oracle DataServer broker whose environment variables you want to set.
3. Click the Oracle DataServer broker that you want to view. The **Oracle DataServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Oracle DataServer Configuration** page appears.
5. Click **Edit**.
6. Click **Environment Variables**.

7. Enter each variable name and then provide its value in the format `name=value`.
8. Click **Save** when you finish adding the variables and their values.

Edit an Oracle DataServer broker configuration

You can edit an Oracle DataServer configuration from the management console.

To edit an Oracle DataServer broker configuration:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the Oracle DataServer broker whose configuration you want to edit.
3. Click the Oracle DataServer broker. The **Oracle DataServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Oracle DataServer Configuration** page appears.
5. Click **Edit**. (For details about the broker or server properties, see Oracle DataServer broker properties on page 59 and Oracle DataServer server properties on page 63.)
6. Make the configuration changes, and then click **Save**.

Create an Oracle DataServer broker

OpenEdge Management and OpenEdge Explorer provide a predefined default Oracle DataServer configuration; however, you can create a new configuration by modifying the default settings.

To create an Oracle DataServer broker:

1. From the drop-down for **Resources** on the management console menu, click **New OpenEdge Resource** > **Oracle DataServer**. The **Oracle DataServer Configuration** page appears.
2. Type the name of the new Oracle DataServer in the field provided.
3. Click **Save**. The **Oracle DataServer Configuration** page appears, allowing you to configure the broker's properties.

Delete an Oracle DataServer broker

You can delete an inactive Oracle DataServer broker from the management console.

If a broker is running, you cannot delete it. (The **Delete** button is not available.) You must first stop the broker and then delete it.

To delete an Oracle DataServer broker:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the Oracle DataServer broker you want to delete.
3. From the Oracle DataServer brokers list, click the broker that you want to delete. The **Oracle DataServer** details page appears.
4. Click **Delete**, and then click **OK** to confirm the deletion.

Start or stop an Oracle DataServer broker

You can start or stop an Oracle DataServer broker from the management console.

To start or stop an Oracle DataServer broker, refer to the *Starting or Stopping OpenEdge resources* section in *OpenEdge Management: Servers, DataServers, Messengers, and Adapters*.

Start an Oracle DataServer broker automatically

You can set a Oracle DataServer broker to start automatically when the AdminServer starts.

To start an Oracle DataServer broker automatically:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the Oracle DataServer brokers.
3. From the search list, click the Oracle DataServer broker that you want to start automatically. The **Oracle DataServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **Oracle DataServer Configuration** page appears.
5. Click **Edit**.
6. Select the **Auto start** check box.
7. Click **Save**.

View the status of an Oracle DataServer broker

You can view the status of any active Oracle DataServer broker, server groups, and auxiliary processes through the management console. Most of the broker instances provide a status view, enabling you to monitor the status of multiple brokers simultaneously.

To see the status of a broker, the broker must be running or have been running.

To view the status of an Oracle DataServer broker:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the Oracle DataServer broker you want to start automatically.
3. Click the Oracle DataServer broker. The **Oracle DataServer** details page appears.
4. In the **Operations views** section of the page, click **Status**.

   A Status summary and summary details appear and provide the information described in Oracle DataServer broker property Status summary and details on page 67.

Oracle DataServer broker property Status summary and details

Table 25: Oracle DataServer broker property Status summary

Property	Description
Host	The broker's host machine.
Broker Name	The name of the broker whose status you are viewing.
Operating Mode	For DataServer brokers, the operating mode is state-aware.
Broker Status	The current state of the broker.
Broker Port	The TCP/IP port number that the broker listens to.
Broker PID	The process ID of the broker.
Active Servers	The number of active servers.
Busy Servers	N/A
Locked Servers	N/A
Available Servers	N/A
Active Clients (now, peak)	N/A
Client Queue Depth (cur, max)	The number of clients waiting for agents to become available to service their request. The current value (**cur**) represents the number of waiting clients at the moment the status is displayed, and the maximum value (**max**) represents the largest number of clients waiting concurrently since the broker was started.
Total Requests	N/A
Rq Wait (max, avg)	N/A
Rq Duration (max, avg)	N/A

Table 26: Oracle DataServer broker Status details

Property	Description
PID	The process id of the server process
State	The current state of the server process
Port	The TCP/IP port number used by the server
nRq	N/A

Property	Description
nRcvd	N/A
nSent	N/A
Started	The time stamp indicating when the server process is started
Last Change	The time stamp indicating when the server process last changed state

View the Oracle DataServer broker log files

You can view the log files for an Oracle DataServer broker by using the log file viewer. The log file viewer allows you to examine the Oracle DataServer log files through an HTML interface.

To access the log file viewer:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the Oracle DataServer broker whose log file viewer you want to access.
3. From the list, click the Oracle DataServer broker. The **Oracle DataServer** details page appears.
4. In the **Command and control** section of the page, Click **Log File Viewer of Broker** or **Log File Viewer of Servers**.

   You can use the Log File Viewer in the following ways:

   - Use the **Show** field to control how many database log file entries display at one time. The number entered into the **Show** field cannot be less than 10.
   - Use the **Overlap** field to control how many entries are repeated from screen to screen.

   Note: The value in the **Overlap** field cannot be more than the number in the **Show** field minus one. For example, if you show 30 entries, you can overlap only 29 or fewer of them.

   - Click **Reload** after changing the values in either the **Show** field or the **Overlap** field. If you do not reload, the viewer continues to display the previous values.
   - Click **Go To** to control which numbered entry in the log file the viewer display begins. For example, a value of 10 entered into the **Go To** field will begin the display from the tenth log file entry.

   Note: You must click **Go To** after entering a value in the **Go To** field or the viewer will not update its display.

   The default display of entries is in ascending order; choose **Descending** to change the display. Note that the **Show** field dictates the number of entries shown, regardless of whether they display in ascending or descending order.

   - Click **First** to display the first **x** entries, where **x** is the value in the **Show** field.
   - Click **Previous** to display the previous **x** entries, where **x** is the value in the **Show** field.
   - Click **Next** to display the next **x** entries, where **x** is the value in the **Show** field.
   - Click **Last** to display the last **x** entries, where **x** is the value in the **Show** field.

- To view additional log file entries without changing your current starting log file entry, leave the **Go To** field blank, change the value in the **Show** field, and click **Reload**.
- If the contents of the log file have changed since you opened the viewer, the log file viewer indicates this in the **Log file status** field.
- OpenEdge Management and OpenEdge Explorer consider a viewer that has been inactive for more than four hours stale and at that point release ninety-five percent of any memory being held. If you try to use a stale viewer, OpenEdge Management and OpenEdge Explorer automatically reload the file. Because additional resource activity might have occurred during the viewer's inactivity, the reloaded log file view might not match the previous log file view of that resource.
- OpenEdge Management and OpenEdge Explorer consider a viewer that has been inactive for forty-eight hours dead. Once a viewer dies, OpenEdge Management and OpenEdge Explorer release all of its memory. To return to the log file displayed in a dead view, you must renavigate to it, even if you pinned up the view or saved a link to it before the viewer died.

MS SQL Server DataServer configuration and administration

The OpenEdge® DataServer for MS SQL Server allows the OpenEdge Application Development Environment (ADE) and applications created with OpenEdge in Windows to access Microsoft SQL Server.

You can use OpenEdge Explorer to configure and administer the configuration of an OpenEdge DataServer for SQL Server. The DataServer for SQL Server installation provides one predefined DataServer broker (mssbroker1). You can use this predefined component as a starting point for creating and configuring additional DataServer brokers. Each broker is referred to as an instance.

Use the MS SQL Server DataServer broker

To work with the MS SQL Server DataServer, double-click the **MSS DataServer** folder from the OpenEdge Explorer list frame. There is one pre-defined default MS SQL Server DataServer (mssbroker1).

You can perform the following MS SQL Server actions:

- Configure an MS SQL Server DataServer broker on page 70
- Edit an MS SQL Server DataServer broker configuration on page 76
- Create an MS SQL Server DataServer broker on page 76
- Start or stop an MS SQL Server DataServer broker on page 77
- Start an MS SQL Server DataServer broker automatically on page 77
- View the MS SQL Server DataServer log files on page 79
- Delete an MS SQL Server DataServer broker on page 77
- View the MS SQL Server DataServer log files on page 79

Configure an MS SQL Server DataServer broker

You can view or modify the following configuration properties of an MS SQL Server DataServer from the management console:

- Broker
- Server
- Environment variables

You can also create a new MS SQL Server DataServer broker, view a broker's status, or delete a broker you no longer need.

Each of these tasks is described in the following sections.

View or modify MS SQL Server DataServer broker properties

You can view or modify MS SQL Server DataServer Broker properties in the following categories: general properties, owner information, controlling NameServer information, Data Service, setting logging properties, and advanced features.

To view or modify broker properties for an MS SQL Server broker:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the MS SQL DataServer broker whose properties you want to view or modify.
3. Click the Edit icon. The **MSS DataServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **MSS DataServer Configuration** page appears. From this page, you can:
   - View a read-only display of the MS SQL Server broker properties as described in MS SQL Server DataServer broker properties on page 70.
   - Click **Edit** to modify the broker properties. For details, see Edit an MS SQL Server DataServer broker configuration on page 76.

MS SQL Server DataServer broker properties

The following topics describe the MS SQL Server DataServer broker properties.

MS SQL Server DataServer broker General properties

Table 27: MS SQL Server DataServer broker General properties

Property	Description
Working directory	Your general working directory. The default value is `@{WorkPath}`.
Operating mode	For DataServer brokers, the operating mode is state-aware. The default value is state-aware.

Property	Description
Port number	The type of the TCP/IP listening port that the DataServer broker uses. The default value is 4446.
Auto start	Starts the DataServer automatically when the AdminServer starts. The **Auto start** check box is cleared by default.

MS SQL Server DataServer broker Owner Information properties

Table 28: MS SQL Server DataServer broker Owner Information properties

Property	Description
Group name	(UNIX only) The name of the group.
Username	Your user name. The default is blank.
Password	Your password. The default is blank.

MS SQL Server DataServer broker Controlling NameServer properties

Table 29: MS SQL Server DataServer broker Controlling NameServer properties

Property	Description
Register with NameServer	A check box that indicates whether to register the DataServer with a controlling NameServer.
Controlling NameServer	Identifies the NameServer that the DataServer uses to find a broker that supports the correct application services. (If you did not select to register the DataServer with a NameServer, the field is dimmed and unavailable.)

Property	Description
Registration mode	How the broker specifies its hostname when registering with its controlling NameServer. This hostname information is passed onto a client application when it attempts to connect to a Data Service that the broker supports. The choices are: • **Register IP** — (Default) This setting is the most efficient mechanism, and can be used in most cases. It registers with the IP address of the machine where the broker is located. • **Register-LocalHost** — The broker registers with the hostname of the machine that it runs on. Use this setting when the broker runs on a machine with a single hostname and more than one IP address. • **Register-HostName** — The broker registers with the values specified in the hostName property. Use this setting when your clients need a fully qualified host name to connect to a broker in a different DNS domain.
Registration host name	(UNIX only) The name of the group.

MS SQL Server DataServer broker Data Service properties

Table 30: MS SQL Server DataServer broker Data Service properties

Property	Description
Data service name list	Data services are services that are provided by a selected broker's servers. The default service name is the broker name. The default value is blank.
Supports default service	A check box that indicates whether the MS SQL Server broker supports the default service. Select the check box to have the broker support the default service, or click to clear the check box and cancel support for the default service. The **Supports default service** check box is cleared by default.

MS SQL Server DataServer broker Logging Setting properties

Table 31: MS SQL Server DataServer broker Logging Setting properties

Property	Description
Broker log filename	The general broker log filename. The default is `@{WorkPath}\mssbroker1.broker.log`.

Property	Description
Broker logging level	The amount of information written to the broker log. The possible values are as follows: • **None** — No log file is written. • **Error Only** — Only error-related information is written to the log file, which reduces the size of the log file. • **Basic** — Only some information is written to the log file. • **Verbose** — All broker activity is written to the log file. This might produce large files that use up disk space. • **Extended** — Information in addition to **Verbose**.
Append to broker log file	Create a new broker log file each time a broker is started, even if the broker log file specified already exists. The **Append to broker log file** check box is selected by default.
Broker logging entry types	A single entry or comma delimited list of logging entry types.
Broker log file threshold size	Specifies a limit on how big the log file can get (in bytes). When the log is full, the broker creates a new log file with a sequence number.
Maximum number of broker log files	Limit on how many broker log files will be kept on the system when there is a threshold limit. It represents the number of log files including the current on and includes any rolled over log files that already exist when the process started.

MS SQL Server DataServer broker Advanced Features properties

Table 32: MS SQL Server DataServer broker Advanced Features properties

Property	Description
Maximum client instances	The maximum number of client connections that the broker can support concurrently. The default value of 512 that is set for this parameter is high enough to ensure that the number of client connections is unlimited. Realistically, however, the system-level resources needed to support an unlimited number of clients might be exhausted before this limit is reached. If you see broker log entries that contain Exception Messages for "out of space" or "OutOfMemory" you may need to reduce the maximum number of client connections to a more reasonable value. Note, however, that lowering this value can cause some client requests to be rejected ("Exceeded Max Clients"). Configuring and starting multiple brokers to handle higher client loads alleviates the problem.

Property	Description
Priority weight (0-100) (dynamic property)	An integer value between 0 and 100 that influences the share of the workload that the selected broker receives. The larger the value, the heavier the load that is distributed to the broker. Using the **Priority weight** value, the NameServer distributes client requests to the same DataServer service across all DataServer brokers. The default value is 0. You can update this property dynamically. Any changes will affect all current and new brokers.
Registration retry (in seconds)	After registering with its controlling NameServer, the DataServer broker periodically sends "keep-alive" messages to the NameServer to let the NameServer know that the broker is still active. The **Registration retry** value is the number of seconds that pass between "keep-alive" messages. The default value is 30.

View or modify MS SQL Server DataServer server properties

You can view or modify MS SQL Server DataServer properties in the following categories: **General** and **Logging Setting**.

To view or modify server properties for an MS SQL DataServer broker:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the MS SQL DataServer broker whose server properties you want to view or modify.
3. From the list, click the MS SQL DataServer broker. The **MSS DataServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **MSS DataServer Configuration** page appears. From this page, you can:
   - View a read-only display of the MS SQL Server server properties as described in MS SQL Server DataServer server properties on page 74.
   - Click **Edit** to modify the server properties. For details, see Edit an MS SQL Server DataServer broker configuration on page 76.
   - View the MS SQL Server DataServer log file. For details, see View the MS SQL Server DataServer log files on page 79.

MS SQL Server DataServer server properties

The following topics describe the MS SQL Server DataServer server properties.

MS SQL Server DataServer server General properties

Table 33: MS SQL Server DataServer server properties

Property category or property	Description
Server executable file (dynamic property)	The pathname of the DataServer executable that the broker runs. The default for the DataServer for MS SQL Server in Windows is: `@{Startup\DLC}bin\_msssrv.exe` You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.
Server startup parameters (dynamic property)	The startup parameters for the server. You should not modify the default parameters, which are: `-svub -S -X -N TCP -U X -P x -hs 0 -s 40` You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.
PROPATH	A list of directories OpenEdge searches to find procedures.
Minimum port number (dynamic property)	The minimum port number that you want to make available for each DataServer broker. The default is blank. You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.
Maximum port number (dynamic property)	The maximum port number that you want to make available for each DataServer broker. The default is blank. You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.

MS SQL Server DataServer server Logging Setting properties

Table 34: MS SQL Server DataServer server Logging Setting properties

Property	Description
Server log filename	The DataServer log filename. Enter a valid path name. Installed Default: `@{WorkPath}\mssbroker1.server.log`

Set MS SQL Server DataServer broker environment variables

You can set environment variables for the DataServer broker for MS SQL Server. You can set one or more environment variables that:

- Are accessible from your ABL application
- Affect the environment in which your application runs

For the MS SQL Server DataServer, there are no required environment variables in Windows.

To set MS SQL Server DataServer environment variables:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the MS SQL DataServer whose environment variables you want to set.
3. From the list, click the MS SQL DataServer broker. The **MSS DataServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **MSS DataServer Configuration** page appears.
5. Click **Edit**.
6. Click **Environment Variables**.
7. Enter each variable name and then provide its value in the format `name=value`.
8. Click **Save** when you finish adding the variables and their values.

Edit an MS SQL Server DataServer broker configuration

You can edit an MS SQL Server DataServer broker configuration from the management console.

To edit an MS SQL Server DataServer configuration:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the MS SQL DataServer whose configuration you want to edit.
3. From the list, click the MS SQL DataServer broker. The **MSS DataServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **MSS DataServer Configuration** page appears.
5. Click **Edit**. (For details about the broker and server properties, see MS SQL Server DataServer broker properties on page 70 and MS SQL Server DataServer server properties on page 74.)
6. Make the configuration changes, and then click **Save**.

Create an MS SQL Server DataServer broker

OpenEdge Management and OpenEdge Explorer provide a predefined default MS SQL Server DataServer configuration; however, you can create a new configuration by modifying the default settings.

To create an MS SQL Server DataServer broker:

1. From the drop-down for **Resources** on the management console menu, click **New OpenEdge Resource** > **MSS DataServer**. The **MSS DataServer Configuration** page appears.
2. Type the name of the new DataServer in the field provided.
3. Click **Save**. The **MSS DataServer Configuration** page appears, allowing you to establish the broker's properties.

Delete an MS SQL Server DataServer broker

You can delete an inactive DataServer broker for MS SQL Server from the management console.

If a broker is running, you cannot delete it. You must first stop the broker and then delete it.

To delete an MS SQL Server DataServer broker:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the MS SQL DataServer you want to delete.
3. From the list, click the MS SQL DataServer broker. The **MSS DataServer** details page appears.
4. Click **Delete**, and then click **OK** to confirm the deletion.

Start or stop an MS SQL Server DataServer broker

You can start or stop an MS SQL Server DataServer broker from the management console.

To start or stop an MS SQL DataServer broker, refer to the Starting or Stopping OpenEdge resources section in *OpenEdge Management: Servers, DataServers, Messengers, and Adapters*

Start an MS SQL Server DataServer broker automatically

You can set a broker for MS SQL Server to start automatically when the AdminServer starts.

To start an MS SQL DataServer broker automatically:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the MS SQL DataServer you want to start automatically.
3. MS SQL DataServer broker. The **MSS DataServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **MSS DataServer Configuration** page appears.
5. Click **Edit**.
6. Select the **Auto start** option by clicking in the check box.
7. Click **Save**.

View the status of an MS SQL Server DataServer broker

You can view the status of any active MS SQL Server DataServer broker, server groups, and auxiliary processes through the management console. Most of the broker instances provide a status view, enabling you to monitor the status of multiple brokers simultaneously.

To see the status of a broker, the broker must be running or have been running.

To view the status of an MS SQL Server DataServer broker:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the MS SQL DataServer broker whose status you want to view.
3. From the list, click MS SQL DataServer broker. The **MSS DataServer** details page appears.
4. In the **Operations views** section of the page, click **Status**.

A Status summary and summary details appear and provide the information described in MS SQL Server DataServer broker property status summary and details on page 78.

MS SQL Server DataServer broker property status summary and details

Table 35: MS SQL Server DataServer broker property status summary

Property	Description
Host	The broker's host machine.
Broker Name	The name of the broker whose status you are viewing.
Operating Mode	For DataServer brokers, the operating mode is state-aware.
Broker Status	The current state of the broker.
Broker Port	The TCP/IP port number that the broker listens to.
Broker PID	The process ID of the broker.
Active Servers	The number of active servers.
Busy Servers	N/A
Locked Servers	N/A
Available Servers	N/A
Active Clients (now, peak)	N/A

Property	Description
Client Queue Depth (cur, max)	The number of clients waiting for agents to become available to service their request. The current value (**cur**) represents the number of waiting clients at the moment the status is displayed, and the maximum value (**max**) represents the largest number of clients waiting concurrently since the broker was started.
Total Requests	N/A
Rq Wait (max, avg)	N/A
Rq Duration (max, avg)	N/A

Table 36: MS SQL Server DataServer broker status details

Property	Description
PID	The process id of the server process
State	The current state of the server process
Port	The TCP/IP port number used by the server
nRq	N/A
nRcvd	N/A
nSent	N/A
Started	The time stamp indicating when the server process is started
Last Change	The time stamp indicating when the server process last changed state

View the MS SQL Server DataServer log files

You can view the log files for an MS SQL Server DataServer by using the log file viewer. The log file viewer allows you to examine the MS SQL Server DataServer log files through an HTML interface.

To access the log file viewer:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the MS SQL DataServer broker whose log file viewer you want to access.
3. From the list, click the MS SQL DataServer broker. The **MSS DataServer** details page appears.
4. In the **Command and control** section of the page, Click **Log File Viewer of Broker** or **Log File Viewer of Servers**. You can use the Log File Viewer in the following ways:

   - Use the **Show** field to control how many database log file entries display at one time. The number entered into the **Show** field cannot be less than 10.

- Use the **Overlap** field to control how many entries are repeated from screen to screen.

  Note: The value in the **Overlap** field cannot be more than the number in the **Show** field minus one. For example, if you show 30 entries, you can overlap only 29 or fewer of them.

- Click **Reload** after changing the values in either the **Show** field or the **Overlap** field. If you do not reload, the viewer continues to display the previous values.
- Click **Go To** to control which numbered entry in the log file the viewer begins its display with. For example, a value of 10 entered into the **Go To** field will begin the display from the tenth log file entry.

  Note: You must click **Go To** after entering a value in the **Go To** field or the viewer will not update its display.

  The default display of entries is in ascending order; choose **Descending** to change the display. Note that the **Show** field dictates the number of entries shown, regardless of whether they display in ascending or descending order.

- Click **First** to display the first **x** entries, where **x** is the value in the **Show** field.
- Click **Previous** to display the previous **x** entries, where **x** is the value in the **Show** field.
- Click **Next** to display the next **x** entries, where **x** is the value in the **Show** field.
- Click **Last** to display the last **x** entries, where **x** is the value in the **Show** field.
- To view additional log file entries without changing your current starting log file entry, leave the **Go To** field blank, change the value in the **Show** field, and click **Reload**.
- If the contents of the log file have changed since you opened the viewer, the log file viewer indicates this in the **Log file status** field.
- OpenEdge Management and OpenEdge Explorer consider a viewer that has been inactive for more than four hours stale and at that point release ninety-five percent of any memory being held. If you try to use a stale viewer, OpenEdge Management and OpenEdge Explorer automatically reload the file. Because additional resource activity might have occurred during the viewer's inactivity, the reloaded log file view might not match the previous log file view of that resource.
- OpenEdge Management and OpenEdge Explorer consider a viewer that has been inactive for forty-eight hours dead. Once a viewer dies, OpenEdge Management and OpenEdge Explorer release all of its memory. To return to the log file displayed in a dead view, you must renavigate to it, even if you pinned up the view or saved a link to it before the viewer died.

4

Configure NameServers

You can configure NameServer property settings and add new NameServer instances in OpenEdge Management and OpenEdge Explorer. You can also start, stop, and delete NameServer instances, as well as view their status and log files.

For details, see the following topics:

- NameServer configuration and administration
- Configure a NameServer
- Edit a NameServer configuration
- Create a NameServer
- Delete a NameServer
- Configure fault tolerance and load balancing
- Start or stop a NameServer
- View the status of a NameServer
- Handle UDP and firewall issues
- View the NameServer log file

NameServer configuration and administration

You can optionally use the NameServer to mediate client connections for instances of OpenEdge Unified Broker products that you configure with OpenEdge Management or OpenEdge Explorer. The Unified Broker products include all OpenEdge resources that are managed through the AdminServer. For more information on the Unified Broker products and the AdminServer, see *OpenEdge Getting Started: Installation and Configuration*.

When you choose to use the NameServer, each component instance registers with only one NameServer, which is known as the instance's controlling NameServer. Each time an instance starts up, it registers with its controlling NameServer by sending its location and other configuration information. The NameServer uses this information to help resolve client connection requests. A particular NameServer can accept registrations from any number of instances, including instances of more than one type of component.

Depending on the way you configure and use the OpenEdge product, you can decide to specify a controlling NameServer to handle client connection requests. For example, your configuration might be simple and not require the location transparency the NameServer provides, or it might be more complex and, therefore, dependent on this and other NameServer features.

The OpenEdge Unified Broker products that you can optionally choose to register with the NameServer include:

- AppServers
- AppServer Internet Adapters
- DataServers for Oracle and MS SQL Server configured to use the NameServer
- SonicMQ Adapters
- WebSpeed Transaction Servers

NameServer features

You can use OpenEdge Management or OpenEdge Explorer to configure both local NameServer instances and remote NameServer instances. The NameServer installation provides one sample NameServer (NS1), and you can use the sample as a template for creating and configuring additional NameServers.

In its role as a connection arbitrator, the NameServer provides location transparency, connection-level fault tolerance, server-level fault tolerance, and optional load balancing. For details about these features, see Configure fault tolerance and load balancing on page 92.

Use with the NameServer

To work with a NameServer instance, select the **NameServer** folder from the management console's list pane. There is one pre-defined default NameServer (NS1).

You can perform the following NameServer actions:

- Configure a NameServer on page 83
- Edit a NameServer configuration on page 90
- Create a NameServer on page 91
- Start or stop a NameServer on page 93
- Delete a NameServer on page 91

- Configure fault tolerance and load balancing on page 92
- View the status of a NameServer on page 94
- Handle UDP and firewall issues on page 95
- View the NameServer log file on page 96

NameServer as an optional client connection request handler

Configuring a NameServer to work with the AppServer, a DataServer, a SonicMQ Adapter, or a WebSpeed Transaction Server is optional. Depending on the way you configure and use the OpenEdge product, you can decide if you want to specify a controlling NameServer. For example, your configuration might be simple and not require location transparency or load balancing, or it might be more complex and, therefore, dependent on those features.

Local and remote NameServers

Before configuring a NameServer, you must know the network location where it is to run. For any network host, you can configure two types of NameServer instances:

- **Local** — An instance that runs locally on the host where it is defined
- **Remote** — An instance that references a NameServer that is defined and runs on a machine that is remote from the host where the reference is defined

The purpose of defining a remote NameServer instance is to provide an easy way for having multiple Unified Broker products refer to a controlling NameServer that runs on another machine. You cannot start, stop, obtain status, or modify any properties of a remote NameServer instance other than the host name and port number that it references.

Configure a NameServer

The OpenEdge® NameServer controls a pool of brokers that register (with the NameServer) the application services they provide. The NameServer can then direct client connection requests to a broker that supports a requested application service. The NameServer also provides location transparency, and, when the NameServer Enterprise Edition is installed, the ability to perform load balancing. Load balancing allows you to balance client workload among multiple brokers that service the same application (that is, the same set of procedures and resources).

For more information, see Configure local instances on page 83 and Configure remote NameServer instances on page 89.

Configure local instances

You configure a local NameServer to run on the selected host where it is defined. Unified Broker products running on the same host as a local NameServer instance can use it as their controlling NameServer.

You can view or edit the configuration properties of a NameServer from the management console. You can also create a new NameServer, view a NameServer's status, delete a NameServer you no longer need, or view the NameServer's log file.

To configure a local NameServer instance:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the local NameServer broker instance you want to configure.
3. Click the NameServer broker instance. The **NameServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **NameServer Configuration** page appears. From this page, you can:
   - View a read-only display of the NameServer properties as described in NameServer properties on page 84.
   - Click **Edit** to modify the NameServer properties. For details, see Edit a NameServer configuration on page 90.

NameServer properties

The following topics describe the NameServer properties.

NameServer Location properties

Table 37: NameServer Location properties

Property	Description
Location	A read-only property that indicates whether the NameServer is local or remote: • **Local** — The NameServer runs locally on the selected host. You can configure all the properties of a local NameServer. • **Remote** — The NameServer runs remotely on a network machine that is separate from the selected host. You can set this property only when you first create the NameServer instance. To change the location of a NameServer, you must first delete and then recreate the NameServer with the new location setting.
Host name	The name of the host. If the NameServer is local, this is set to **localhost**. Otherwise, type the name of the remote host where the NameServer is to run.
Port number	The number of the UDP port that the NameServer uses to listen for client connection requests and registration messages from AppServers, DataServers, and WebSpeed Transaction Servers.

NameServer General properties

Table 38: NameServer General properties

Property	Description
Working directory	The NameServer working directory. Enter the pathname.
Broker keep alive timeout(dynamic property)	A value, in seconds, that indicates how often the NameServer checks for Unified Broker instances that have timed out. When an OpenEdge Unified Broker instance registers with a NameServer, the instance indicates how often it sends "keep-alive" messages by setting a registration retry property value (a property setting in Advanced Features for the Unified Broker). Once a NameServer determines that it did not receive a "keep-alive" message from a Unified Broker instance within the broker's registration retry time, the NameServer automatically unregisters the instance. Note that you should use a **Broker keep alive timeout** value that is somewhat larger than the Unified Broker instance's registration retry value. The NameServer adjusts the specified value to allow for normal networking delays that can occur within your computer network. To keep the NameServer from using up computer resources unnecessarily, set the **Broker keep alive timeout** to a value that is at least 30% larger than the typical Unified Broker registration retry value. You can update this property dynamically. Any changes will affect all current and new NameServers.
Auto start	A check box that you can select if you want the NameServer to start automatically when the controlling AdminServer starts.
jvmargs	Any jvm arguments you set.

NameServer Logging Setting properties

Table 39: NameServer Logging Setting properties

Property	Description
Server log filename	The NameServer log filename. Enter a valid path name. The installed default is `@{WorkPath}\NS1.NS.log`.

Property	Description
NameServer logging level (dynamic property)	A value that specifies the amount of information that is written to the NameServer log. Select from the following values in the drop-down list: • **None** — Log no entries. This is equivalent to turning logging off. • **Error Only** — Log OpenEdge error messages. This includes all error messages and is unrelated to the entry types specified. Errors continue to be logged at all higher levels. • **Basic** — Logging entry types determine the logged information. Each entry type generates at least some output. This is the default. • **Verbose** — Logging entry types determine the logged information, but it is typically more information than **Basic**. • **Extended** — Logging entry types determine the logged information, but it is typically more information than **Verbose**. The installed default is **Basic**. You can update this property dynamically. Any changes will affect all current and new NameServers.
Append to NameServer log file	A check box that indicates whether a new server log file should be created when the NameServer is started. To create a new NameServer log file each time the NameServer is started, even if the NameServer log file already exists, clear the check box. To append log entries to the existing NameServer log file, select the check box. **Append to NameServer log file** is selected by default.

Property	Description
Logging entry types (dynamic property)	The default and only supported entry is NSPlumbing. Specifying the NSPlumbing log entry type turns on logging for different NameServer actions based on logging level: • **Basic** — Logs messages when a NameServer starts and stops, for load balancing status, and when registering and unregistering brokers. This is the default logging level. • **Verbose** — Same as **Basic**, plus logs requests from clients and status information on the communication between client and NameServer and between NameServer and brokers. • **Extended** — Same as **Verbose**, plus logs debugging messages on broker unregistering. Log entries generated for this type contain the tag NSPlumbing.You can update this property dynamically. Any changes will affect all current and new NameServers.
Log file threshold size	A value of 0 or a value between 500,000 and 2,147,483,647, where 0 means there is no limit on the log file size other than what the operating system imposes.
Maximum number of log files	The number of rolled-over log files to keep. The value can be 0 or a value between 2 and 999999, where 0 means there is no limit on the number of NameServer log files to keep. The specified number represents the maximum total number of log files to keep on disk at any time, including the current log file being written to. When the file becomes equal to or greater than the **Log file threshold size**, the client process renames it and creates a new log file. The file is renamed as follows, where ###### is a number starting at 000001 and increasing to 999999, after which it rolls back over to 000001:`filename.######.extension`

NameServer Advanced Features properties

Table 40: NameServer Advanced Features properties

Property	Description
Neighboring NameServers	A list of selected NameServers to which this NameServer can forward connection requests for application services that are not registered with it (that is, the application service name is unknown). The list contains all NameServer instances defined in the management console, initially with none selected. To make a NameServer a neighboring NameServer, select the unselected NameServer to highlight it. You can select as many NameServer instances as you want to be neighboring NameServers. Each neighboring NameServer thus appears highlighted in the list. To remove a NameServer from the list of neighboring NameServers, select the already-selected NameServer to remove the highlight. The setting of this property is optional. When a NameServer receives a request for an application service name that is unknown and there are no neighboring NameServers specified, the NameServer sends a message to the requesting client indicating that the application service is unknown. When such a NameServer has neighboring NameServers specified, it forwards the request to each of the neighboring NameServers. The first neighboring NameServer that has the application service name, and responds to the connection request, provides the specified connection to the client.
TCP/IP Version	Internet Protocol for network communication: • **IPv4** — Accepts only IPv4 connections • **IPv6** — Allows IPv4 and IPv6 connections using mapped address (where supported) The default is IPv4. You must also set the **jvmArgs** property in the `ubroker.properties` file to override the default behavior of the JVM for IPv6 connections. `jvmArgs=-Djava.net.preferIPv4Stack=false -Djava.net.preferIPv6Addresses=true`
Enable dynamic property updates	Allows dynamic changes to occur to several of the instance's properties, without requiring you to shut down the instance. This property is disabled by default.

Property	Description
	Note: Any property that is dynamic is labeled as such in this document.

NameServer Environment Variables properties

Table 41: NameServer Environment Variables properties

Property	Description
Environment	Allows you to add and delete environment variables. Environment variables that you add to the list are scoped to the process in which the NameServer executes.

Configure remote NameServer instances

A remote NameServer instance is a reference to a NameServer that runs on some other host. By defining a remote instance, you allow a Unified Broker running on the local host to use the NameServer running on the remote host as its controlling NameServer. Thus, a remote NameServer configuration requires that you define:

- One local NameServer instance on the host where the NameServer runs
- One remote instance on each host where a Unified Broker uses the remote NameServer as its controlling NameServer

For example, consider that you are creating a NameServer instance on one system, which is named Sys1. The NameServer instance is actually physically running under an AdminServer on another system, which is named Sys2. The instance is remote for Sys1 and local for Sys2.

To create a remote instance, you need the host name and port number of Sys2 and its NameServer. You then use that information to create a remote NameServer on Sys1.

To configure a remote NameServer instance:

1. Configure at least one local NameServer instance on the remote host.
2. Using the management console, connect to the AdminServer on a host where you want to define the remote instance, and filter or search for the **NameServer** broker instances for that host.
3. Click the Edit icon for an existing remote NameServer instance. The **NameServer** details page appears.

   You can also create a new NameServer, specifying its name and indicating that its location is remote. For details, see Create a NameServer on page 91.

4. In the **Command and control** section of the details page, click **Configuration**. The **NameServer Configuration** page appears.
5. Click **Edit**. The **Location** default is **local**. If you are editing an existing instance or creating a new instance, change the location to **remote**.
6. Set or change the host name to a machine where you have a NameServer defined as a local instance. Set the port number to the port on which the local instance you want to reference is listening.
7. Click **Save**.

Edit a NameServer configuration

You can edit a NameServer configuration from the management console.

To edit a NameServer configuration:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the NameServer broker instance whose configuration you want to edit.
3. Click the NameServer broker instance. The **NameServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **NameServer Configuration** page appears.
5. Click **Edit**. (For details about the NameServer properties, see NameServer properties on page 84.)
6. Make the configuration changes, and then click **Save**. Any changes you make are saved in the `ubroker.properties` file.

Set NameServer environment variables

You can set or delete environment variables for a NameServer.

To set NameServer environment variables:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the NameServer broker instance whose environment variables you want to set.
3. Click the NameServer broker instance. The **NameServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **NameServer Configuration** page appears.
5. Click **Edit**.
6. Click **Environment Variables**.
7. Enter each variable name and then provide its value in the format `name=value`.
8. Click **Save** when you finish adding the variables and their values.

Delete NameServer environment variables

To delete NameServer environment variables:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the NameServer broker instance whose environment variables you want to delete.
3. Click the NameServer broker instance. The **NameServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **NameServer Configuration** page appears.
5. Click **Edit**.

6. Click **Environment Variables**.
7. Delete the variable and value pair (or pairs).
8. Click **Save** when you finish.

Create a NameServer

OpenEdge Management and OpenEdge Explorer provide a sample configuration NameServer instance (NS1) to use as a default.

You can create two types of NameServer instance:

- **Local** — Where the instance definition resides on, and the NameServer itself executes on, the selected host.
- **Remote** — Where the instance definition is a reference to a NameServer that resides and executes on a machine that is remote from the selected host.

To create a NameServer:

1. From the drop-down for **Resources** on the management console menu, click **New OpenEdge Resource** > **NameServer**. The **NameServer Configuration** page appears.
2. Type the name of the new NameServer in the field provided.

   Note: The NameServer name is case sensitive and can include any character except a period (.) or square brackets ([ ]). The name must be unique among all configured NameServer names.

3. Click **Save**. The **NameServer Configuration** page appears, allowing you to configure the NameServer's properties. (For details about the properties, see NameServer properties on page 84.)

   Once you create a NameServer instance, you cannot change the location setting unless you delete and recreate the instance.

   Note: If you are defining a remote NameServer instance, set its host name and port number properties to reference a NameServer that is defined as a local NameServer on some other machine in your network. For details, see Configure remote NameServer instances on page 89.

Delete a NameServer

You can delete any NameServer instance from the management console as long as neither of the following conditions is true:

- The NameServer instance is running.
- One or more Unified Broker instances reference this NameServer instance as their controlling NameServer.

To delete the NameServer, you must stop the NameServer instance and delete all Unified Broker instances that reference it.

To delete a NameServer instance:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the NameServer broker instance you want to delete.
3. Click the NameServer broker instance. The **NameServer** details page appears.
4. Click **Delete**, and then click **OK** to confirm the deletion.

   The **OpenEdge Management Resources** page appears. A set of next-step options that are related to resources are available for your use.

Configure fault tolerance and load balancing

With the NameServer, you have access to connection-level fault tolerance, server-level fault tolerance, and load balancing.

Connection-level fault tolerance

Multiple NameServers can cooperate to resolve a client connection. If one NameServer becomes unavailable, another can take its place, thus providing connection-level fault tolerance. When you configure and use NameServers with connection-level fault tolerance, each requesting client uses the Unified Broker connection provided by the first responding NameServer and ignores the rest.

You can configure two types of connection-level fault tolerance:

- NameServer replication
- NameServer neighbors

NameServer replication

To use NameServer replication, you first configure multiple NameServers on one subnet to listen on the same UDP port. You then have each Unified Broker instance register with its controlling NameServer and have each client contact its connecting NameServer using the subnet broadcast address and the same UDP port number. Thus, Unified Broker registrations are automatically replicated in each NameServer, and client connection requests are handled by each NameServer. The first replicated NameServer to respond to a client request provides the Unified Broker connection.

NameServer neighbors

To use NameServer neighbors, you configure multiple NameServers on one or more subnets. For each such NameServer, you specify the instance name of each of the other NameServers that you want to act as a NameServer neighbor. When a client makes a connection request to a specific NameServer, if this NameServer determines that it cannot satisfy the request (usually because it does not recognize the specified application service), it passes the request to all of its NameServer neighbors. The first NameServer neighbor to respond to a client request provides the Unified Broker connection.

You can configure and use NameServer replication and NameServer neighbors together to provide the maximum in connection-level fault tolerance.

Server-level fault tolerance

If you have the load-balancing option installed, you can configure multiple Unified Broker instances of a given type to support the same application service and share the same controlling NameServer. In this configuration, the controlling NameServer distributes client connection requests randomly across Unified Broker instances. If one Unified Broker instance becomes unavailable to handle client requests, another is available to take its place, thus providing server-level fault tolerance.

Load balancing

If you specify a priority weight factor for each Unified Broker instance, the NameServer distributes the connection requests proportionately, thus balancing the load. For example, suppose that an Inventory service is supported by WebSpeed brokers WS1, WS2, and WS3, which have weight factors of 20, 20, and 60, respectively. The selection algorithm used by the NameServer guarantees that WS1 and WS2 are selected 20% of the time and WS3 is selected 60% of the time.

Note: You do not assign application services or weight factors to NameServers, but rather to Unified Broker instances. Each Unified Broker instance passes its weight factor and list of application service names to its controlling NameServer when it registers with the NameServer.

Location transparency

When a Unified Broker instance is started, it registers with its controlling NameServer the application services that it supports. (An application service is an arbitrary designation for the business function that a Unified Broker instance provides.) The NameServer maintains a separate list of application services for each Unified Broker type. When a client of a certain type requests a connection to an application service, the NameServer redirects the connection request to a Unified Broker instance of the corresponding type that supports the requested application service. Because the client always connects indirectly to a Unified Broker through the NameServer, the client does not need to know the location of the Unified Broker instance. It needs to know only the location of the NameServer and the name of a valid application service that it wants.

Start or stop a NameServer

From the management console, you can directly start or stop a local NameServer on the machine where it runs. You can also set a NameServer to start automatically when the AdminServer starts. For details, see Start a NameServer automatically on page 93.

Note: To start or stop a remote NameServer, access the corresponding local NameServer instance selected in the AdminServer of the host on which it runs.

To start or stop a NameServer, refer to the *Starting or Stopping OpenEdge resources* topic in *OpenEdge Management: Servers, DataServers, Messengers, and Adapters*.

Start a NameServer automatically

You can set a NameServer to start automatically when the AdminServer starts.

To start a NameServer automatically:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the NameServer broker instance you want to start automatically.
3. Click the NameServer broker instance. The **NameServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **NameServer Configuration** page appears.
5. Click **Edit**.
6. Click the **General** tab.
7. Select the **Auto start** option by clicking in the check box.
8. Click **Save**.

View the status of a NameServer

You can view the status of any running NameServer instance through the management console.

Each NameServer instance provides a separate status view. You can monitor the status of only those NameServer instances running locally on a selected host.

To view the status of a NameServer instance:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the NameServer broker instance whose status you want to view.
3. Click the NameServer broker instance. The **NameServer** details page appears.
4. In the **Operations views** section of the page, click **Status**.

A status summary and summary details appear.

The summary, as described in NameServer status summary properties on page 94, provides collective property and statistics information for the NameServer and the OpenEdge Unified Broker instances that it coordinates.

The details, as described in NameServer status detail properties on page 95, provide a table of status information for all Unified Broker instances that are registered with the NameServer. Each row in the table contains information for a single Unified Broker instance.

NameServer status summary properties

Table 42: NameServer status summary properties

Property or statistic	Description
Host	The name of the host machine
Port	The UDP port number that the NameServer listens to
Timeout	The Unified Broker keep alive timeout value for the NameServer

Property or statistic	Description
Start time	When the NameServer started
Number of AppServices	The number of application services that the NameServer is coordinating
Number of Brokers	The number of Unified Broker instances the NameServer is coordinating
Total client requests received	The number of client requests the NameServer has received
Total client requests rejected	The number of client requests the NameServer has rejected

NameServer status detail properties

Table 43: NameServer status detail properties

Property	Description
AppService Name	The name of an application service registered with the NameServer.
UUID	The unique number for the Unified Broker instance.
Name	The Unified Broker instance name.
Host	The name of the host where the Unified Broker instance resides.
Port	The TCP/IP port number used by the Unified Broker instance to handle client connection requests.
Weight	The priority weight assigned to the Unified Broker instance for load balancing.
Timeout	The number of seconds that must pass between keep-alive messages sent by the Unified Broker instance before the NameServer assumes that the instance is no longer available. Once the timeout expires, the NameServer automatically unregisters the Unified Broker instance.

Handle UDP and firewall issues

Whenever an AppServer Internet Adapter (AIA) or WebSpeed Messenger instance communicates with a NameServer, the instance is assigned a UDP port. The NameServer uses this port to send a response message to the instance. By default this port is dynamically assigned by the operating system within the UDP port range of 1024 to 65535.

If there is a firewall between the instance and the NameServer, the default setting requires that all UDP ports in that range must be open from the network where the NameServer exists to the network where the AIA or Messenger instance exists. Otherwise, communication between the instance and the NameServer cannot occur.

However, the only port that must be open in the firewall is the controlling NameServer port for UDP traffic going from the network containing the AIA or Messenger instance to the network containing the NameServer. To increase security, you can limit the range of ports from which an AIA or Messenger instance can be assigned an UDP port. The NameServer Client Port Minimum (minNSClientPort) and the NameServer Client Port Maximum (maxNSClientPort) properties can be used to reduce the number of UDP ports that you must open in the firewall. When you set these properties, OpenEdge chooses the first port in that range that is not currently used by some other application for UDP broadcast communications. If no port in this range is available, the default behavior is to retry the ports for a period of time.

View the NameServer log file

The NameServer log file viewer allows you to examine the contents of log files through an HTML interface.

For details, see the topic about using the NameServer log file viewer in *OpenEdge Management: Servers, DataServers, Messengers, and Adapters*.

5

Configure AppServers and AppServer Internet Adapters

You can use OpenEdge Management or OpenEdge Explorer to configure the properties of OpenEdge resources that you are monitoring, such as AppServers or AppServer Internet Adapters. Any configuration changes that you make to either of these resources in OpenEdge Management or OpenEdge Explorer are automatically reflected in the configuration property file `ubroker.properties`.

The following topics provide information about working with AppServer and AppServer Internet Adapter configuration properties.

For details, see the following topics:

- AppServer configuration and administration
- For additional AppServer information
- AppServer Internet Adapter configuration and administration

AppServer configuration and administration

With the AppServer, you can make ABL applications available to multiple machines over a network.

You can use OpenEdge Management or OpenEdge Explorer to configure new AppServer instances and administer the configurations of existing instances. The AppServer installation provides one sample AppServer (asbroker1). You can use this sample component as a template for creating and configuring additional AppServers. Each AppServer is called an instance.

Overview of AppServer configuration

Each AppServer consists of an AppServer broker and one or more AppServer agents. An AppServer agent is a single ABL Virtual Machine (AVM) instance running on an AppServer. Each AppServer agent executes ABL procedures on behalf of AppServer clients, including ABL and Open Clients (Java, .NET, and Web services clients).

An AppServer can support clients using two different session models, which are dependent on its operating mode:

- **Session-managed** — Includes stateless, state-aware, and state-reset operating modes. These operating modes maintain a single session and connection between the AppServer and the client for all client requests.
- **Session-free** — Includes only the state-free operating mode. This operating mode maintains no session or connection between the AppServer and the client. In this operating mode, the AppServer treats any request from a client as independent from all other requests. As a result, some SESSION handle attributes in the ABL application have no valid meaning, or they provide different information.

The AppServer supports these session models for supported AppServer clients.

For more information, see *OpenEdge Application Server: Developing AppServer Applications*.

AppServer features

The AppServer broker handles the following activities:

- Optionally registers with a NameServer the application services that it provides for access by clients
- Manages connections between clients and a pool of AppServer agents
- Maintains the status of each AppServer agent in its pool and dynamically scales the number of servers according to changing demand
- When configured for stateless or state-free operating mode, dispatches remote requests to AppServer agents
- When configured for state-reset or state-aware operating mode, connects each client to a dedicated AppServer agent that handles all remote requests for that client until the connection is terminated

AppServer actions

To work with an AppServer instance, double-click the **AppServer** folder in the management console's list frame. There is one pre-defined default AppServer (asbroker1).

You can perform the following AppServer actions:

- Configure an AppServer on page 100
- Create an AppServer on page 119
- Start and stop an AppServer on page 119
- Add or trim AppServer agents on page 120
- View the status of an AppServer on page 121
- View AppServer log files on page 126

AppServer configuration properties

An AppServer starts up by using the configuration information stored in the `ubroker.properties` file. The file contains the property settings for all server resources on the host where OpenEdge is installed.

The ubroker.properties file

The `ubroker.properties` file consists of a hierarchical structure of configuration entities, in which parent entities provide configuration information that you can override or extend in each child entity. Each configuration entity has a name that begins the entity definition, and the definition contains configuration settings for one or more product instances.

For example, property settings for an instance of an AppServer are determined hierarchically, as shown in the following table.

Table 44: Ubroker.properties file structure

Configuration entity name	Configuration entity function
[UBroker]	Defines default property settings for various OpenEdge brokers, including all AppServer brokers.
[UBroker.AS]	Defines default property settings for all instances of an AppServer.
[UBroker.AS.*product-instance-name*]	Defines property settings for a particular instance of an AppServer. The `ubroker.properties` file can contain several of these entities, each with a unique *`product-instance-name`*. **Note:** The *`product-instance-name`*is unique and case-sensitive. It can contain alphanumeric and special characters, except period (“.”), colon (":"), or square brackets (“[ ]”). The name must not contain blank spaces.

Parent entities provide default values for all of their child entities. However, at any child level a redefinition of any value supersedes the default value of its parent. All children, from their definition level down, inherit this new value.

Like the `ubroker.properties` file, a similar file, `conmgr.properties`, stores all the properties for each instance of an OpenEdge database. For more information about OpenEdge Management and OpenEdge Explorer and the database configuration properties, see Configure OpenEdge Databases on page 19.

The `ubroker.properties` file resides in the properties subdirectory of the OpenEdge install directory. The file is editable and any changes that you make to it are automatically reflected in the management console.

Note: Although making manual edits to the `ubroker.properties` file is possible, Progress Software recommends that you use the Mergeprop utility or the management console to make property changes. For more information about the Mergeprop utility, see *OpenEdge Getting Started: Installation and Configuration*.

Configure an AppServer

From the management console, you can view or modify an AppServer broker's default configuration properties in the following overall categories:

- Broker
- Agent
- TLS
- Messaging
- Environment variables

You can also create a new AppServer instance, view an instance's status, delete an instance you no longer need, or view the instance's log files.

Each of these tasks is described in the sections that follow.

View or modify AppServer broker properties

You can view or modify AppServer broker properties in the following categories: general properties, owner information, controlling NameServer information, AppService Name List information, logging setting details, and advanced features.

To view or modify AppServer broker properties:

1. From the OpenEdge Management console menu, click **Resources**. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer broker instance whose properties you want to view or modify. For example, type the default AppServer broker name `asbroker1` in the filter field or select AppServer as the selection in the **Type** drop-down.
3. From the list of AppServer brokers, click the AppServer broker whose properties you want to view or modify. The **AppServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **AppServer Configuration** page appears with the **Broker** properties in focus. From this page, you can:
   - View a read-only display of the AppServer broker properties as described in AppServer broker properties on page 100.
   - Click **Edit** to modify the broker properties,. For details, see Edit an AppServer configuration on page 119.

AppServer broker properties

AppServer broker General properties

Table 45: AppServer Broker General properties

Property	Description
Auto start	Indicates whether the server automatically starts when the controlling AdminServer starts.

Property	Description
Port number	The number of the TCP/IP port that the server broker listens on.
Working directory	Your working directory. The default is the working directory set during OpenEdge installation.
Operating mode	The operating mode for this server.

AppServer broker Owner Information properties

Table 46: AppServer Broker Owner Information properties

Property	Description
Username	The username of an account that has system-administrative rights.
Group name	(UNIX only) The name of the group.
Password	The username account password in Windows.
Password Confirm	Confirmation of the username account password. This field appears only if you are modifying the properties. Note that this field does not appear if the AdminServer is running on a non-Windows host machine.

AppServer broker Controlling NameServer properties

Table 47: AppServer Broker Controlling NameServer properties

Property	Description
Register with NameServer	Indicates whether or not to register the AppServer with a controlling NameServer.
Controlling NameServer	The name of the NameServer with which this AppServer registered. If you did not select to register the broker with a NameServer, the field is disabled.

Property	Description
Registration mode	How the broker specifies its hostname if it is registering with a controlling NameServer. This hostname information is passed onto a client application when it attempts to connect to an application service that the AppServer broker supports. The choices are: • **Register-IP** — (Default) This setting is the most efficient mechanism and can be used in most cases. It registers with the IP address of the machine where the broker is located. • **Register-LocalHost** — The broker registers with the hostname of the machine that it runs on. Use this setting when the broker runs on a machine with a single hostname and more than one IP address. • **Register-HostName** — The broker registers with the values specified in the HostName property. Use this setting when your clients need a fully qualified host name to connect to a broker in a different DNS domain.
Registration host name	The host name alias of the local system.

AppServer Broker AppService Name List properties

Table 48: AppServer Broker AppService Name List properties

Property	Description
Application service names	A list to which you can add or delete application service names
Supports default service	Indicates whether or not the AppServer broker supports the default service

AppServer Broker Logging Setting properties

Table 49: AppServer Broker Logging Setting properties

Property	Description
Broker log filename	The broker log filename.
Broker logging level (dynamic property)	A value that specifies the amount of information that is written to the broker log. The following values are available: • **None** — Log no entries. This is equivalent to turning logging off. • **Error Only** — Log OpenEdge error messages. This includes all error messages and is unrelated to the entry types specified. Errors continue to be logged at all higher levels.

Property	Description
	• **Basic** — Broker logging entry types determine the logged information. Each entry type generates at least some output. This is the default. • **Verbose** — Logging entry types determine the logged information, but it is typically more information than **Basic**. • **Extended** — Logging entry types determine the logged information, but it is typically more information than **Verbose**. You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.
Append to broker log file	Indicates whether or not a new broker log file should be created when the AppServer is started.

Broker logging entry types (dynamic property)	A comma-separated list of the following valid entry types for agent logging:
	• **UBroker.AutoTrim** — Logs messages about automatic agent thread trimming by the broker based on the **Auto trim timeout** property. Information is logged at logging level **Basic** and higher.
	• **UBroker.Basic** — Logs messages about broker and agent startup and shutdown at the following logging levels:
	  • **Basic** — Logs NameServer registration and connections from clients.
	  • **Verbose** — Logs information about all the property values read from the `ubroker.properties` file for that broker instance, and more information on the connection from clients.
	• **UBroker.ThreadPool** — Logs messages about the pool of threads managed by the broker. These threads are used to control the client requests and the agent processes. This entry type logs messages when adding, removing, and communicating with these threads. Information is logged at logging level **Basic** and higher.
	You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.
	Turn on the following logging entry types only at the request of Progress Software Corporation Technical Support:
	• **UBroker.ClientFSM** — Logs messages about the state of the client thread as it processes the requests, from the moment it receives a connection request to the end of the request. Information is logged at logging level **Basic** and higher.
	• **UBroker.ServerFSM** — Logs messages about the state of the agent thread as it processes the requests, from the moment it receives a connection request to the end of the request. Information is logged at logging level **Basic** and higher.

Property	Description
	• **UBroker.ClientMsgStream** — Logs messages about messages exchanged between the broker and client threads in binary format. Information is logged at logging level **Basic** and higher. • **UBroker.ServerMsgStream** — Logs messages about messages exchanged between the broker and agent threads in binary format. Information is logged at logging level **Basic** and higher. • **UBroker.ClientMsgQueue** — Logs messages about the client thread as it processes queued messages. Information is logged at logging level **Basic** and higher. • **UBroker.ServerMsgQueue** — Logs messages about the agent thread as it processes queued messages. Information is logged at logging level **Basic** and higher. • **UBroker.ClientMemTrace** — Traces messages being processed by the client threads. Information is logged at logging level **Basic** and higher. • **UBroker.ServerMemTrace** — Traces messages being processed by the agent threads. Information is logged at logging level **Basic** and higher. • **UBroker.Stats** — Logs statistics about the number of requests, plus maximum, minimum, and average request wait and duration times. Information is logged at logging level **Basic** and higher. • **UBroker.All** — Logs all Unified Broker entry types. You can specify a separate logging level for each log entry type and use a wildcard (*) to specify multiple entry types. For example: `UBroker.Basic:3,UBroker.ThreadPool:4, UBroker.Client*` For details, see the information about client logging in *OpenEdge Deployment: Startup Command and Parameter Reference*.

Property	Description
Broker log file threshold size	A value of 0 or a value between 500,000 and 2,147,483,647, where 0 means there is no limit on the log file size other than what the operating system imposes.
Maximum number of broker log files	The number of rolled-over log files to keep. The value can be 0 or a value between 2 and 999999, where 0 means there is no limit on the number of broker log files to keep. The specified number represents the maximum number of log files to keep on disk at any time, including the current log file being written to. When the file becomes equal to or greater than the **Broker log file threshold size**, the client process renames it and creates a new log file. The file is renamed as follows, where `######` is a number starting at 000001 and increasing to 999999, after which it rolls back over to 000001: *`filename.######.extension`*

AppServer Broker Advanced Features properties

Table 50: AppServer Broker Advanced Features properties

Property	Description
Maximum client instances	The maximum number of client connections that the broker can support concurrently. The default is 512, which is high enough to ensure that the number of client connections is virtually unlimited. Realistically, however, the system-level resources needed to support an unlimited number of clients might be exhausted before this limit is reached. If you see broker log entries that contain Exception Messages for "out of space" or "OutOfMemory" you might need to reduce the maximum number of client connections to a more reasonable value. Note, however, that lowering this value can cause some client requests to be rejected ("Exceeded Max Clients"). Configuring and starting multiple brokers to handle higher client loads alleviates the problem.
Priority weight (0-100) (dynamic property)	An integer value between 0 and 100 that influences the share of the workload that the selected AppServer receives. The larger the value, the heavier the load that is distributed to the server. The NameServer distributes client requests across all AppServers that have the same application service in proportion to the **Priority weight** value. The default is 0. You can update this property dynamically. Any changes will affect all current and new brokers.

Property	Description
Registration retry (in seconds)	After registering with its controlling NameServer, the AppServer periodically sends "keep-alive" messages to the NameServer to let the NameServer know that it is still active. The **Registration retry** value is the number of seconds that pass between "keep-alive" messages. The default is 30.
Server startup timeout (dynamic property)	Specifies the amount of time, in seconds, that the broker waits for an active agent to become available before starting a new agent process. The default is 3. You can update this property dynamically. Any changes will affect only new agents that start after the property value has change; existing (running) brokers and agents are not updated.
Request timeout (dynamic property)	Specifies the amount of time, in seconds, that the broker will wait for an agent to become available for processing a request. The broker waits for this period only if the **Maximum servers** setting has been reached. After the timeout has expired, the client receives a "no servers available" error message. The default is 15. You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.
Auto trim timeout (dynamic property)	The amount of time, in seconds, that the broker waits before automatically trimming the number of running agents. The broker keeps track of the maximum number of agents that are simultaneously busy during the interval you specify. At the end of the interval, the broker attempts to trim the number of agents to match the **Maximum servers** for the interval, or the **Minimum servers**, whichever is greater. You can disable this feature by setting the timeout to zero (0). The default **Server Auto-trim Interval** is 1800 seconds. Progress Software recommends that you use this default setting to avoid unnecessary process management that might result from frequent starting and trimming of agents. You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.

Property	Description
TCP/IP version	Internet Protocol for network communication: • **IPv4** — Accepts only IPv4 connections • **IPv6** — Allows IPv4 and IPv6 connections using mapped address (where supported) The default is IPv4. You must also set the jvmArgs property in the `ubroker.properties` file to override the default behavior of the JVM for IPv6 connections. `jvmArgs=-Djava.net.preferIPv4Stack=false -Djava.net.preferIPv6Addresses=true.`
jvmargs	Java System Properties.
Enable dynamic property updates	Allows dynamic changes to occur to several of the instance's properties, without requiring you to shut down the instance. This property is disabled by default. **Note:** Any property that is dynamic is labeled as such in this document.
AppServer Keepalive	Allows the AppServer to recognize that a client bound to it is no longer connected.
serverASK Activity Timeout	Specifies the number of seconds with no activity that the AppServer waits before sending the client a message requesting that the connection be maintained. The property is used only if the AppServer Keepalive property is set to **allowServerASK**.
serverASK Response Timeout	Specifies the number of seconds with no activity after the AppServer sends the client a message (requesting that the connection be maintained) before the client is disconnected. This property is used only if the AppServer Keepalive property is set to **allowServerASK**.
Enable debugging through the broker (dynamic property)	Specifies whether or not the debugger is enabled; select the check box to enable it. If you disable this property while the debugger is connected, the debugger connection terminates. **Note:** The **Broker debugger port number** property must also be set to a valid value for the debugger to be properly enabled. This property is disabled by default.

Property	Description
Broker debugger port number(dynamic property)	Specifies the port number that the broker debugger will use when it starts. You can choose any valid value; if the port you specify is already in use, the debugger service will fail to start. If you change the port number while the debugger service is active, the debugger service will be shut down and restarted using the new port number The default port number is 3099.
Enable SSL for broker debugger (dynamic property)	Specifies whether the broker debugger should accept only TLS-encrypted connections. If you enable this property, the Progress Developer Studio debugger will be required to establish an TLS session. If you enable or disable the property while the debugger service is active, the debugger will be disconnected. The debugger service will be shut down and then restarted using the new selection. This option is disabled by default.
Use the broker's private key/digital certificate for debugger(dynamic property)	When enabled, indicates that, when using TLS, the broker debugger service should use the same private key as used by the AppServer agents. If this property is enabled, then the **Debugger private key/digital certificate alias name** and **Password to access the debugger's key/certificate** property settings are ignored. This property is enabled by default.
Debugger private key/digital certificate alias name (dynamic property)	The name of the private key that should be used for the debugger service when the agent's private key is not being used. This property must be defined if **Enable SSL for broker debugger** is enabled.
Password to access the debugger's key/certificate(dynamic property)	The password required to access the private key specified by **Debugger private key/digital certificate alias name**. The default for this property is blank..
Broker debugger passphrase (dynamic property)	An optional property that represents the passphrase required by the debugger to connect to the debugger service in the broker. If the debugger is connected to the debugger service when the passphrase is changed, the debugger remains connected. The next time the debugger attempts to connect to the debugger service, the new password is required. The default for this property is blank.

Property	Description
Broker debugger passphrase Confirm	Confirms the passphrase provided in **Broker debugger passphrase**.
Publish directory	The name of the local directory where you want r-code to be placed when you are publishing from Progress Developer Studio. If you are publishing remotely from Progress Developer Studio, you must set this property.

View or modify AppServer Agent properties

You can modify the following Agent properties for the AppServer Broker: general properties, logging setting details, pool range, and advanced features.

To view or modify AppServer Agent properties:

1. From the OpenEdge Management console menu, click **Resources**. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer broker instance whose properties you want to view or modify. For example, for the default AppServer broker name, type `asbroker1` in the filter field or select AppServer from the **Type** drop-down.
3. From the list of AppServer broker instances, click the AppServer broker whose properties you want to view or modify. The **AppServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **AppServer Configuration** page appears with the **Broker** properties in focus.
5. Click the **Agent** tab. From this page, you can do the following:
   - View a read-only display of the AppServer Agent properties as described in AppServer Agent properties on page 109.
   - Click **Edit** to modify the broker properties. For details, see Edit an AppServer configuration on page 119.

AppServer Agent properties

AppServer Agent General properties

Table 51: AppServer Agent General properties

Property	Description
Server executable file (dynamic property)	Either the default agent executable pathname or a different agent executable file of your own choosing. You generally need to specify a nondefault value only if you generated a new executable using the OEBUILD utility. You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.

Property	Description
Server startup parameters (dynamic property)	The OpenEdge startup parameters that you want to specify to start each agent. You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated. **Note:** It is recommended to set the Stack size startup parameter value as -s 128 in the **Server startup parameters** field, when working with databases having large schemas.
PROPATH	The search path that agents use to locate ABL procedures that they execute. Specify a list of directory pathnames in the form of an OpenEdge PROPATH. Make sure that you copy all of your AppServer procedures to one or more of the directories in the PROPATH. You can enter the pathname directly.
Minimum port number (dynamic property)	The minimum TCP/IP port number from a range that each agent process can listen on. When each agent starts, it allocates a port that is not being used from the specified range up to the **Maximum port number** setting. You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.
Maximum port number (dynamic property)	The maximum TCP/IP port number from a range that each agent can listen on. When each agent starts, it allocates a port that is not being used from the specified range down to the **Minimum port number** setting. You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.
Flush statistical data (dynamic property)	How often the data is flushed by the agent to the broker. This is expressed as the number of remote procedure calls between flushes. You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.

AppServer Agent Logging Setting properties

Table 52: AppServer Agent Logging Setting properties

Property	Description
Server log filename	The log file name.

Property	Description
Server logging level (dynamic property)	A value that specifies the amount of information to be written to the agent log. Each logging level name has the indicated numeric value. You can select from the following values in the drop-down list: • **None** — Log no entries. This is equivalent to turning logging off. • **Error Only** — Log OpenEdge error messages. This includes all error messages and is unrelated to the entry types specified. Errors continue to be logged at all higher levels. • **Basic** — Server logging entry types determine the logged information. Each entry type generates at least some output. This is the default. • **Verbose** — Logging entry types determine the logged information, but it is typically more information than **Basic**. • **Extended** — Logging entry types determine the logged information, but it is typically more information than **Verbose**. You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.

Server logging entry types (dynamic property)	A comma-separated list of valid entry types for agent logging:
	• **4GLTrace** — Turns on the logging for the execution of the following ABL statements: RUN, FUNCTION (user-defined functions), PUBLISH, and SUBSCRIBE. Logs information at **Basic** or higher.
	• **ASDefault** — Combines the ASPlumbing and DB.Connects log entry types. This is the default value for AppServer agents.
	• **ASPlumbing** — Turns on logging for different actions depending on the logging level specified:
	  • **Basic** — Logs messages when an AppServer agent starts and stops, and when client processes connect and disconnect from the AppServer agent.
	  • **Verbose** — Same as **Basic**, plus logging of the execution of the top level procedure (that is, the procedure that the client process asks the AppServer agent to execute on its behalf, but not any procedure that the top level procedure executes).
	  • **Extended** — Same as **Verbose**, plus messages about agent state, and status messages about read and write socket operations. Do not turn this on unless directed to do so by Progress Software Technical Support.
	• **DB.Connects** — Turns on logging of database connections and disconnections. The messages include database name and user ID number. Logs information at logging level **Basic** or higher.

Property	Description

	• **DynObjects.DB** — Turns on logging for dynamic database-related objects (for example, TEMP-TABLE or DATASET) as they are created and destroyed. Logs information at **Basic** or **Verbose**.
	• **DynObjects.XML** — Turns on logging for dynamic XML-related objects (for example, x-document or x-noderef) as they are created and destroyed. Logs information at level **Basic** or **Verbose**.
	You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.
	• **DynObjects.Other** — Turns on logging for dynamic objects that do not match the other dynamic object categories (for example, procedure or server socket) as they are created and destroyed. Logs information at **Basic** or **Verbose**.
	• **ProEvents.UI.Char** — Turns on logging of keystroke events for printable characters. Logs information at **Basic** or higher.
	• **ProEvents.UI.Command** — Turns on logging of keystroke events for nonprintable characters at the following logging levels:
	  • **Basic** — Logs nonprintable keystrokes and a subset of Windows GUI events
	  • **Verbose** — Logs nonprintable keystrokes and all Windows GUI events
	• **ProEvents.Other** — Turns on logging of COM, asynchronous, and server socket events. Logs information at **Basic** or higher, beginning with the events for which you have written triggers.
	• **QryInfo** — Turns on database query information logging at the following logging levels, with the same amount of information logged for each level:
	  • **Basic** — Logs static queries and initial dynamic queries.
	  • **Verbose** — Logs static queries and multiple instances of dynamic queries.
	• **SAX** — Turns on logging for various stages of execution using the ABL SAX parser.
	You can specify a separate logging level for each log entry type and use a wildcard (*) to specify multiple entry types. For example:
	`4GLTrace:3, ASPlumbing:2,DynObjects.*`
	For details, see the information about client logging in *OpenEdge Deployment: Startup Command and Parameter Reference*.

Append to server log file	Indicates whether or not a new agent log file should be created when the AppServer is started.

Property	Description
Server log file threshold size	A value of 0 or a value between 500,000 and 2,147,483,647, where 0 means there is no limit on the agent log file size other than what the operating system imposes.
Maximum number of server log files	The number of rolled-over log files to keep. The value can be 0 or a value between 2 and 999999, where 0 means there is no limit on the number of agent log files to keep. The specified number represents the maximum total number of log files to keep on disk at any time, including the current log file being written to. When the file becomes equal to or greater than the **Server log file threshold size**, the client process renames it and creates a new log file. The file is renamed as follows, where `######` is a number starting at 000001 and increasing to 999999, after which it rolls back over to 000001:*`filename.######.extension`*
Server watch dog interval	The interval (in seconds) for the server log file watchdog thread. If the **Server log file threshold size** property is specified, the broker will roll over the server log file when its size is larger than the logThreshold value.

AppServer Agent Pool Range properties

Table 53: AppServer Agent Pool Range properties

Property	Description
Initial number of servers to start	The number of agents you want the broker to start initially.
Minimum servers	The minimum number of agents before the broker starts additional agents. If you trim the number of agents below this value, the server starts any additional agents needed to maintain the specified minimum when the next client connects.
Maximum servers	The maximum number of agents that this server can have running at the same time.

AppServer Agent Advanced Features properties

Table 54: AppServer Agent Advanced Features properties

Property	Description
4GL debugger enabled	Indicates if the AppServer remote debugging facility is enabled. If the facility is enabled, ABL client applications have the ability to step into remote AppServer procedures and WebSpeed procedures have the ability to step into remote AppServer procedures.

Property	Description
Activate procedure (dynamic property)	The name of an ABL procedure that executes before the AppServer executes a remote procedure request for a client connection that is unbound. **Activateprocedure** is available only for an AppServer running in stateless operating mode. You can update this property dynamically. Any changes will affect all current and new agents.
Deactivate procedure (dynamic property)	The name of an ABL procedure that executes after the AppServer executes a remote procedure request for a client connection that is unbound. The **Deactivateprocedure** property is available only for an AppServer running in stateless operating mode. You can update this property dynamically. Any changes will affect all current and new agents.
Connect procedure (dynamic property)	The name of an ABL procedure that executes when a client attempts to connect to this AppServer. You can update this property dynamically. Any changes will affect all current and new agents.
Disconnect procedure (dynamic property)	The name of an ABL procedure that executes when a client disconnects from this AppServer. You can update this property dynamically. Any changes will affect all current and new agents.
Startup procedure(dynamic property)	The name of an ABL procedure that executes when an AppServer agent starts up for this AppServer. The **Startup procedure** property is not available for an AppServer running in state-reset operating mode. You can update this property dynamically. Any changes will affect only new agents; existing (running) agents are not updated.
Shutdown procedure (dynamic property)	The name of an ABL procedure that executes in an AppServer agent when the AppServer agent shuts down. AppServer agents shut down when you use the trim feature and also when you shut down the AppServer. The **Shutdown procedure** property is not available for an AppServer running in state-reset operating mode. You can update this property dynamically. Any changes will affect all current and new agents.

Property	Description
Startup procedure parameters (dynamic property)	The parameters to pass the startup procedure when it executes. The **Startup procedure parameters** property takes a character string as an input parameter. For example:`DEFINE INPUT PARAMETER startup-data AS CHARACTER NO-UNDO.` You can set this parameter to any arbitrary value. If you do not specify a parameter in this field, the parameter is set to the unknown value (?) when the AppServer agent executes the startup procedure. You can update this property dynamically. Any changes will affect only new agents; existing (running) agents are not updated.
Execution Time Limit	Sets the maximum time in seconds that a remote procedure can execute on a given AppServer. The time limit applies to all remote procedures that execute on the AppServer. The default value is zero, which indicates that no time limit exists.

View or modify AppServer TLS properties

You can view or modify the following TLS properties for an AppServer Broker: general properties and advanced features.

To view or modify AppServer TLS properties:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer broker instance whose TLS properties you want to view or modify.
3. Click the Edit icon. The **AppServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **AppServer Configuration** page appears with the **Broker** properties in focus.
5. Click the **SSL** tab. From this page, you can:
   - View a read-only display of the AppServer TLS properties as described in AppServer TLS properties on page 115. These tables follow this procedure.
   - Click **Edit** to modify the TLS properties. For details, see Edit an AppServer configuration on page 119.

AppServer TLS properties

The following topics describe the AppServer TLS properties.

AppServer TLS General properties

Table 55: AppServer TLS General properties

Property	Description
Enable SSL client connections	If the check box is selected, specifies that all connections to this AppServer must use TLS tunneling.

Property	Description
Private key/Digital certificate alias name	The alias name within the OpenEdge keystore of the private key and digital certificate entry to use to authenticate all connections to this AppServer. If not otherwise selected, the AppServer uses the default_server server certificate alias.
Password to access the key/certificate	The password to use for accessing the private key and digital certificate. **Note:** OpenEdge Explorer encrypts your password in the `ubroker.properties` file. If you set the password in `ubroker.properties` manually, you must specify the password as an encrypted value. You can obtain this value using the genpassword utility located in the bin directory of your OpenEdge installation. If you use the default_server server certificate, it also has a default password that you do not need to specify.
Password to access the key/certificate Confirm	Confirm the password. This field appears only if you are editing the properties.

AppServer TLS Advanced Features properties

Table 56: AppServer TLS Advanced Features properties

Property	Description
Disable SSL session caching	If the check box is selected, caching for the TLS client session is disabled.
SSL session cache timeout	The length of time, in seconds, that an TLS client session is held in the session cache, during which an TLS client can resume its session. The default is 180.

View or modify AppServer Messaging properties

You can view or modify several AppServer Messaging properties.

To view or modify AppServer Messaging properties:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer broker instance whose messaging properties you want to view or modify.
3. Click the Edit icon. The **AppServer** details page appears.

4. In the **Command and control** section of the page, click **Configuration**. The **AppServer Configuration** page appears with the **Broker** properties in focus.
5. Click the **Messaging** tab. From this page, you can:
   - View a read-only display of the AppServer Messaging properties as described in AppServer Messaging properties on page 117.
   - Click **Edit** to modify the AppServer Messaging properties. For details, see Edit an AppServer configuration on page 119.

AppServer Messaging properties

The following table lists the AppServer Messaging properties.

Table 57: AppServer Messaging properties

Property	Description
SonicMQ ServerConnect enabled	Allows the asbroker or wsbroker agent to start a personal SonicMQ Adapter instead of going to a separate one running elsewhere.
SonicMQ ServerConnect broker log filename	A valid path and file name.
Broker logging level (dynamic property)	A value that specifies the amount of information to be written to the SonicMQ ServerConnect broker log. You can select from the following values in the drop-down list: • **None** — Log no entries. This is equivalent to turning logging off. • **Error Only** — Log error messages. This includes all error messages and is unrelated to the entry types specified. Errors continue to be logged at all higher levels. • **Basic** — Broker logging entry types determine the logged information. Each entry type generates at least some output. This is the default. • **Verbose** — Logging entry types determine the logged information, but it is typically more information than **Basic**. • **Extended** — Logging entry types determine the logged information, but it is typically more information than **Verbose**. You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.
Append to broker log file	If the check box is selected, Indicates that a new broker log file should be created, even if the broker log file already exists, when the SonicMQ ServerConnect broker is started.
SonicMQ ServerConnect server log filename	A valid path and file name.

Property	Description
Server logging level (dynamic property)	A value that specifies the amount of information to be written to the SonicMQ ServerConnect server log. Each logging level name has the indicated numeric value. You can select from the following values in the drop-down list: • **None** — Log no entries. This is equivalent to turning logging off. • **Error Only** — Log error messages. This includes all error messages and is unrelated to the entry types specified. Errors continue to be logged at all higher levels. • **Basic** — Broker logging entry types determine the logged information. Each entry type generates at least some output. This is the default. • **Verbose** — Logging entry types determine the logged information, but it is typically more information than **Basic**. • **Extended** — Logging entry types determine the logged information, but it is typically more information than **Verbose**. You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.
Append to server log file	If the check box is selected, indicates that a new server log file should be created, even if the broker log file already exists, when the SonicMQ ServerConnect instance is started.

Set or delete AppServer broker environment variables

You can set or delete environment variables. For the AppServer, environment variables that appear in the list are scoped to:

- The process in which the AppServer broker executes
- Every AppServer agent that the AppServer broker starts

To set or delete AppServer broker environment variables:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer broker instance whose environment variables you want to set.
3. Click the Edit icon. The **AppServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **AppServer Configuration** page appears.
5. Click **Edit**.
6. Click **Environment Variables**.
7. Enter each variable name and then provide its value in the format `name=value`.

   Alternatively, you can remove an existing variable by deleting the name and value pair.

8. Click **Save** when you finish setting (or deleting) the variables and their values.

Edit an AppServer configuration

You can edit the configuration properties for an AppServer resource from the management console.

To edit AppServer configuration properties:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer broker instance whose configuration properties you want to edit.
3. Click the Edit icon. The **AppServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **AppServer Configuration** page appears.
5. Click **Edit**, and make the changes you want to the properties. (For details about the properties, see AppServer broker properties on page 100, AppServer Agent properties on page 109, AppServer TLS properties on page 115, AppServer Messaging properties on page 117 and AppServer Internet Adapter (AIA) properties on page 128.)
6. Click **Save**. Any changes you make are also reflected in the `ubroker.properties` file.

Create an AppServer

OpenEdge Management and OpenEdge Explorer provide a sample AppServer instance (asbroker1) with a default configuration. Each new instance you create uses the same default configuration, which you can edit.

To create an AppServer:

1. From the drop-down for **Resources** on the management console menu, click **New OpenEdge Resource** > **AppServer**.

   The **AppServer Configuration** page appears.

2. Type the name of the new AppServer in the field provided.

   Note: The AppServer name is case sensitive and can include any character except a period (.) or square brackets ([ ]). The name must be unique among all configured AppServer names.

3. Click **Save**. The **AppServer Configuration** page appears, allowing you to configure the AppServer's properties.

Each new AppServer instance you create uses the default configuration. However, the port number must be unique for each AppServer instance in order for the instance to operate properly. Once you create a new instance, you are prompted to change the port number.

Start and stop an AppServer

From the management console, you can start or stop either a local or a remote AppServer. You must first connect to the AdminServer that manages the AppServer you want to start.

You can also choose to start an AppServer automatically when the AdminServer starts.

Note: Prior to starting an AppServer using SonicMQ ServerConnect, you must first configure the AppServer as SonicMQ ServerConnect enabled. To stop a remote AppServer, access the corresponding local AppServer instance selected in the AdminServer of the host where it runs.

To start or stop an AppServer, refer to the *Starting or Stopping OpenEdge resources* topic in *OpenEdge Management: Servers, DataServers, Messengers, and Adapters*.

Start an AppServer automatically

You can set an AppServer to start automatically when the AdminServer starts.

To start an AppServer automatically:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer broker instance you want to start automatically.
3. Click the Edit icon. The **AppServer** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **AppServer Configuration** page appears.
5. Click **Edit**.
6. Select the **Auto start** option by clicking in the check box.
7. Click **Save**.

Add or trim AppServer agents

At run time, you can increase the number of AppServer agents running and available to service AppServer client requests by having the AppServer add a specified number of AppServer agents to its process pool.

You can also trim running AppServer agents, down to the Minimum Servers property setting, from the management console.

More information about trimming AppServer agents

To better understand the use of the Agent Auto-trim feature, consider a scenario involving a broker serving a client/server application running in stateless mode. Assume that the following property values are set:

- Minimum Agent/Server Instances: 10
- Initial Agent/Server Instances to Start: 10
- Maximum Agent/Server Instances: 50
- Agent/Server Auto-trim Interval: 1800

When the broker starts, it starts 10 agents. This also marks the start of the first Agent/Server Auto-trim Interval. Assume that the broker is idle for 30 minutes (it is just before people arrive for work, for instance). At the end of the 30-minute interval, the highest number of agents that were busy at the same time would be zero. (Since there were no requests made of any of the agents, none of the agents was busy.) However, since zero is less than the Minimum Agent/Servers Instances value of 10, no automatic trimming takes place. This marks the start of the second interval.

As the second interval starts, employees begin their workday. There is a sudden demand for agents; in fact, the broker receives so many requests that it must start 20 additional agents. Assume that the high volume of workload, and demand on the agents, continues through the second interval. At the interval's end, the maximum number of busy agents is 30, which is the number the broker scaled up to in response to high demand. Thirty agents are running, and the maximum busy level is 30. No trimming occurs.

During the next interval, the request load diminishes somewhat. At the end of the interval, only 20 agents are concurrently busy. The maximum number of busy agents for the interval remains at 30, since 30 were busy at the start of the interval. Thirty agents are running, and the maximum busy level is 30. No trimming occurs.

Assume that all employees have to attend a company-wide meeting that begins midway through the interval. The highest number of agents busy at that same time was 20. At the end of the interval, there are 30 agents running, with zero agents currently busy. Since the maximum busy level is 20 (at the start of the interval), and the number of running agents is 30 (all are available since everyone is at the company meeting), the broker attempts to trim back the number of agents to 20.

Assume that the company meeting ran for another half-hour, and the entire next interval elapsed without any requests. The maximum busy agents would be zero, with 20 currently running. Since the Minimum Agent/Server Instances property value is set to 10, the broker attempts to trim back to 10 agents.

Keep in mind that the lower you set the Agent/Server Auto-trim Interval value, the more quickly the broker reacts to a drop in the workload. Under a reasonably steady work load, the broker should trim the number of running agents to the number needed. However, if the workload fluctuates a great deal and the Agent/Server Auto-trim Interval is set too low, the broker may trim agents that it must then restart. As stated earlier, repetitive trimming and restarting causes the system to waste resources on unnecessary process management.

Add or trim AppServer agents

To add or trim AppServer agents:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer broker instance to which you want to add or trim agents.
3. Click the Edit icon. The **AppServer** details page appears.
4. If the AppServer is not started, click **Broker Control**. Then click **Start AppServer**.
5. When the AppServer is running, click the backlink for the AppServer broker instance on the breadcrumb trail to return to the details page.
6. In the **Command and control** section of the page, click **Server Pool Control**.
7. Review the **Server pool initial configuration**, the **Servers state**, and the **Server pool summary** information. (You can view detailed information about an agent by clicking on its PID.)
8. To add or trim servers, choose the corresponding option in the **Add**/**Trim** dropdown menu, and type the number of servers in the box provided.
9. Click **Submit**. A confirmation message appears.

View the status of an AppServer

You can view the status of any running AppServer instance through the management console. Status and/or statistical details appear in the following different views:

- The AppServer instance's **Details** page
- The **Broker Control** link on the AppServer instance's **Details** page

- The **Broker PID** link within the instance's **AppServer Control** page
- The **Server Pool Control** link on the AppServer instance's **Details** page

  You can also add and trim AppServer agents from the **Server Pool Control** page

View AppServer instance status from the Details page

To view the status of an AppServer instance from the Details page:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer broker instance whose status you want to view.
3. Click the Edit icon. The **AppServer** details page appears and displays the following information:
   - **Host** — The name of the host machine
   - **Broker** — Whether the broker is active or not running
   - **Operating mode** — One of four possible modes can be reported: **Stateless**, **State-reset**, **State-aware**, and **State-free**
   - **Broker statistics available** — Whether broker statistics are being collected
   - **Servers available** — The number of AppServers running and available to fulfill a connection request from a client to an AppServer through this broker when the broker's status is **ACTIVE**
   - **Should register with NameServer** — The status of **True** or **False** to indicate whether or not the broker resource is registered with a NameServer

View AppServer instance status from the Broker Control link

To view the status of an AppServer instance from the Broker Control link:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer broker instance whose status you want to view.
3. Click the Edit icon. The **AppServer** details page appears.
4. In the **Command and control** section of the page, click **Broker Control**. The **AppServer Control** page for that instance appears and displays the following information:
   - **Broker name** — The name of the selected AppServer broker.
   - **Host** — The name of the host machine.
   - **Port** — The port number.
   - **Broker PID** — The broker's process identification number (PID).
   - **Status** — The running status of the broker. Possible values are:
     - **ACTIVE** — The broker is currently running.
     - **Not Running** — The broker is not currently running.

     The broker can also report **Starting** and **Shutting Down** values; however, depending on the speed of the machine on which your management console is running, you may not see these intermediary states.

- **Operating mode** — One of four possible modes can be reported: **Stateless**, **State-free**, **State-aware**, or **State-reset.**
- **Broker statistics available** — Whether collected broker statistics are available.
- **Enabled** — Whether the broker is enabled.
- **Collect Statistics** — Whether statistics are being collected.

View AppServer instance status from the Server Pool Control link

To view the status of an AppServer instance from the Server Pool Control link:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer broker instance whose status you want to view.
3. Click the Edit icon. The **AppServer** details page appears.
4. In the **Command and control** section of the page, click **Server Pool Control**. The **Server Pool Control** page for that instance appears and displays the following information:
   - **Initial number of servers to start** —The value referenced when the AppServer broker starts AppServers.
   - **Minimum servers** — The minimum number of AppServers that must be simultaneously running before the AppServer broker starts additional servers. The broker strives to maintain this specified minimum. If at any time the number of servers falls below the specified minimum, the broker automatically starts the additional servers needed to maintain the minimum.

     If you set a trim value that requires trimming the number of servers to below the number specified for this field, a message appears.
   - **Maximum servers** — The maximum number of AppServer processes that can be running simultaneously. Add requests you initiate that will exceed the specified maximum are not fulfilled.
   - **Active servers** — The number of AppServers currently running.
   - **Busy servers** — The number of AppServers handling ABL client requests.
   - **Locked servers** — The number of AppServers servicing a bound connection. (This state applies to a stateless AppServer.)
   - **Available servers** — The number of AppServers available to handle broker requests.

A server pool summary provides details about the following:

- **PID** — The process identifier for this AppServer. Click the PID number to display a detail page that provides specific information about this server process and, as necessary, to kill the process. See the section about killing an AppServer process in *OpenEdge Management: Servers, DataServers, Messengers, and Adapters* for more information.
- **State** — The current execution state of the agent.
- **Port** — The TCP/IP port number used by the agent.
- **nRq** — The number of messages sent to the agent.
- **nRcvd** — The number of messages received by the agent.
- **nSent** — The number of requests sent by the agent.
- **CPU Use** — The percentage of CPU user and system time consumed by a process.
- **Memory Use** — The amount of virtual memory (in Kbytes) consumed by a process.

- **Started** — The time stamp indicating when the agent started.
- **Last Change** — The time stamp indicating when the agent last changed execution state.

View AppServer process details

You can access real-time details and statistics that provide you with snapshot information about a running AppServer process at the point you access this information from the **Control** page.

To view process details:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer broker instance to which you want to add or trim agents.
3. Click the Edit icon. The **AppServer** details page appears.
4. In the **Command and control** section of the page, click **Server Pool Control**.
5. Click a **PID** link. The following sections appear:
   - The **Process summary** section identifies the **Process name** and **Process start time**. **User id** and **Group id** values appear when UNIX-based data is shown. The **Parent pid** provides the identifier number associated with the process that spawned this current process.
   - The **Process statistics** section presents details about the database's real-time operational status. Values presented without parentheses identify that the processing time determined since the last scheduled polling interval, as noted, has occurred. Values presented within parentheses have been calculated based on information obtained since the start of the process.

     The following table identifies and describes the fields of information presented in the **Process statistics** section.

     Table 58: Process statistics data

Field	Description
Resident size	The physical size of the process as defined by the host system.
Virtual size	The virtual size of the process as defined by the host system.
CPU	The percentage of time spent using the CPU in either the user or kernel mode since the last scheduled poll.
Weighted CPU	The percentage of time spent using the CPU in either the user or kernel mode since the last scheduled poll divided by the number of CPU processors on the system. This value appears only when there is more than one CPU process on the system where the process is running.
User time	The amount of CPU time spent in the user mode since the last scheduled poll.

Field	Description
Kernel time	The amount of CPU time spent in the kernel mode since the last scheduled poll.
Process time	The sum of the values that appear in the **User time** and **Kernel time** fields.

Stop an AppServer agent

When you stop an AppServer agent, the related process is completed and then stopped.

To stop an AppServer agent:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer broker instance which you want to stop.
3. Click the required instance icon. The **AppServer** details page appears.
4. In the **Command and control** section of the page, click **Server Pool Control**.
5. Click a **PID** link.
6. Click **Stop**. Cick **OK** when you get a confirmation message.

Kill an AppServer agent

When you kill an AppServer agent, the agent is stopped immediately without waiting for any related processes to complete.

To kill an AppServer agent:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer broker instance which you want to kill.
3. Click the required instance icon. The **AppServer** details page appears.
4. In the **Command and control** section of the page, click **Server Pool Control**.
5. Click a **PID** link.
6. Click **Kill**. Cick **OK** when you get a confirmation message.

Delete an AppServer

You can delete any AppServer instance listed in the management console as long as the instance is not running. (The **Delete** button is not available if the instance is running.) If the instance is running, first stop it and then delete it.

To delete an AppServer instance:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter, or search for, the AppServer broker instance you want to delete.
3. Click the AppServer broker instance. The **AppServer** details page appears.
4. Click **Delete**, and then click **OK** to confirm the deletion.

   The **OpenEdge Management Resources** page appears. A set of next-step options that are related to resources are available for your use.

View AppServer log files

The AppServer log file viewer allows you to examine the contents of a log file through an HTML interface.

For details, see the section about using the AppServer log file viewer in *OpenEdge Management: Servers, DataServers, Messengers, and Adapters*.

For additional AppServer information

You can find additional information about using OpenEdge Management with the AppServer in *OpenEdge Management: Servers, DataServers, Messengers, and Adapters*. See that document for details, including specifics about the following:

- Working with the AppServer log file monitor and viewer
- Killing an AppServer process
- Listing AppServer Client connections

AppServer Internet Adapter configuration and administration

With the AppServer Internet Adapter (AIA) you can make AppServer or SonicMQ Adapter application services available over the Web to ABL applications, and you can make AppServer application services available over the Web to .NET and Java Open Clients. The AIA is a Java Servlet that is invoked by a Java Servlet Engine (JSE).

When you use the URL syntax for one of the following connection protocols to connect standard ABL clients (including AppServer and WebSpeed agents, and WebClients) or .NET and Java Open Clients to an AppServer, or to connect standard ABL clients to a SonicMQ Adapter:

- **HTTP** — Directs the client to connect through an AIA instance using an unsecured Internet connection to the Web server that hosts the AIA
- **HTTPS** — Directs the client to connect through an AIA instance using a Transport Layer Security (TLS) Internet connection to the Web server that hosts the AIA

Note: You can also use the AppServer protocol to allow a client to connect directly to an AppServer or a SonicMQ Adapter using the TCP/IP protocol with or without TLS tunneling. This is the preferred option for client access over an intranet.

However, if you want to use a secure Internet connection to the Web server and also secure the connection between the AIA and the AppServer (or SonicMQ Adapter), you can configure the AIA to use TLS for its own connection to the AppServer (or SonicMQ Adapter).

Use the AppServer Internet Adapter

To work with an AIA instance, double-click the **AppServer Internet Adapter** folder from the management console's list pane. There is one pre-defined default AppServer Internet Adapter (Aia1).

The following topics describe these AppServer Internet Adapter actions:

- Configure an AppServer Internet Adapter on page 127
- Edit an AppServer Internet Adapter configuration on page 133
- Create an AppServer Internet Adapter on page 133
- Delete an AppServer Internet Adapter broker on page 135
- View the AppServer Internet Adapter log file on page 134

Configure an AppServer Internet Adapter

You can view or modify the following configuration properties of an AppServer Internet Adapter broker from the management console:

- General
- Controlling NameServer
- Logging Setting
- TLS
- Advanced Features

You can also create a new AppServer Internet Adapter instance, delete an instance you no longer need, or view its log file.

To view or modify AppServer Internet Adapter configuration properties:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer Internet Adapter broker instance whose configuration properties you want to view or modify.
3. Click the Edit icon. The **AppServer Internet Adapter** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **AppServer Internet Adapter Configuration** page appears with the **General** properties in focus. From this page, you can:
    - View a read-only display of the AppServer Internet Adapter instance's properties, as described in AppServer Internet Adapter (AIA) properties on page 128. All of these tables follow this procedure.

- Click **Edit** to modify the properties. For details, see Edit an AppServer Internet Adapter configuration on page 133.

AppServer Internet Adapter (AIA) properties

The following topics describe the AppServer Internet Adapter (AIA) properties.

AIA General properties

Table 59: AIA General properties

Property	Description
Idle connection timeout	Timeout value, in seconds, between an AppServer client and an AIA instance. If a connection is idle for more than the specified timeout value, then the AIA instance automatically disconnects from the AppServer. The default is 3600.
Secure port	Identifies the port on which the Web server listens for requests that specify the HTTPS protocol. This value is used only when the **HTTPS enabled** property check box is selected. If you do not specify a value for this property, and the **HTTPS enabled** property check box is selected, the default value for **Secure port** is 443.
HTTPS enabled	Determines how an AppServer connection using the AIA connects to the AIA's Web server host. If you select this option, clients must use HTTPS to connect and send encrypted data across the Internet to the Web server. If you do not select this option, clients can use HTTP to connect and send data across the Internet without encryption. When the **HTTPS enabled** check box is selected, if an AppServer client uses HTTP to connect to an AppServer, the AIA returns an HTTPS redirection URL to the client. The client automatically retries the connection using the HTTPS URL. The default setting for this check box is enabled.

AIA Controlling NameServer properties

Table 60: AIA Controlling NameServer properties

Property	Description
Connect using NameServer	A check box that indicates whether to connect to an AppServer using a Controlling NameServer.

Property	Description
Controlling NameServer	Identifies the NameServer that the AIA uses to find a broker that supports the correct application services.
Minimum NameServer client port and **Maximum NameServer client port**	Properties used to limit the range of ports of an AIA communicating with the NameServer via UPD. This is important when the AIA and NameServer are separated by a firewall. The value for these two properties must be a number between 1024 and 65535 inclusive (or 0), and **Minimum NameServer client port** must be less than or equal to **Maximum NameServer client port**. If the minimum and maximum values are both set to zero, then a random port number within the implicit fixed range will be dynamically assigned. If both minimum and maximum values are set to the same port number, then this UDP port number will be used when communicating with the NameServer. The default value for both minimum and maximum is 0.
AppServer host name	If you did not select the **Connect using NameServer** check box and the AIA will be directly connected to an AppServer or SonicMQ Adapter, you must specify the AppServer's or SonicMQ Adapter's host location.
AppServer port	If you did not select the **Connect using NameServer** check box and the AIA will be directly connected to an AppServer or SonicMQ Adapter, you must specify the AppServer's or SonicMQ Adapter's port.

AIA Logging Setting properties

Table 61: AIA Logging Setting properties

Property	Description
Log filename	The AIA log filename. Enter a valid path name. The installed default is `@{WorkPath}\Aia1\aia.log`

Property	Description
Logging level	A value that specifies the amount of information to be written to the AIA log. Select from the following values in the drop-down list: • **None** — Log no entries. This is equivalent to turning logging off. • **Error Only** — Log OpenEdge error messages. This includes all error messages and is unrelated to the entry types specified. Errors continue to be logged at all higher levels. • **Basic** — Logging entry types determine the logged information. Each entry type generates at least some output. This is the default. • **Verbose** — Logging entry types determine the logged information, but it is typically more information than **Basic**. • **Extended** — Logging entry types determine the logged information, but it is typically more information than **Verbose**. The default is **Basic**.
Append to log file	A check box that indicates whether or not a new server log file should be created when the AIA is started. To create a new AIA log file each time the AIA is started, even if the AIA log file already exists, clear the **Append to log file** check box. To append log entries to the existing AIA log file, select the **Append to log file** check box. The **Append to log file** check box is selected by default.

Property	Description

Logging entry types	The AIA log entry type turns on logging for different AIA actions based on the logging level. The AIA log entry types provide the following information:
	• **AiaDefault** — Combines AiaMgmt and AiaProp. This is the default entry type for the AIA component.
	• **AiaMgmt** — Logs messages about regular AIA operations, such as AIA version information, instantiation, informational details, and information on the AIA connection pool according to the following logging level settings:
	  • **Basic** — Logs instantiation and informational messages
	  • **Verbose** — Logs the connection pool information
	  • **Extended** — Logs the same information as **Verbose**
	• **AiaProp** — Logs the properties set for the specific AIA instance. These properties are set in the `ubroker.properties` file. Information is logged at **Basic** and higher.
	• **AiaRqst** — Logs information on the GET and POST commands sent by the client and the response back to the client. Information is logged at **Basic** and higher.
	• **AiaUBroker** — Logs information on the communication between the AIA and the Unified Broker (AppServer or SonicMQ Adapter) according to the following Logging level settings:
	  • **Basic** — Provides most of the information
	  • **Verbose** — Logs a binary dump of messages exchanged between the client and the broker/server process

Property	Description
Log file threshold size	A value of 0 or a value between 500,000 and 2,147,483,647, where 0 means there is no limit on the log file size other than what the operating system imposes.
Maximum number of log files	The number of rolled-over log files to keep. The value can be 0 or a value between 2 and 999999, where 0 means there is no limit on the number of AIA log files to keep. The specified number represents the maximum total number of log files to keep on disk at any time, including the current log file being written to. When the file becomes equal to or greater than the **Log file threshold size**, the client process renames it and creates a new log file. The file is renamed as follows, where ###### is a number starting at 000001 and increasing to 999999, after which it rolls back over to 000001:*filename*.######.extension

AIA TLS properties

Table 62: AIA TLS properties

Property	Description
Enable SSL client connections	If the check box is selected, this property directs that the connection to the AppServer or SonicMQ Adapter use TLS tunneling.
Disable client verification of SSL host name	If the check box is selected, this property turns off host verification for an TLS connection. If the check box is cleared, the client compares the host name for the AppServer or SonicMQ Adapter specified by AppServer Location Host Name (or as returned by any Controlling NameServer) with the Common Name specified in the server digital certificate, and raises an error if they do not match. With this parameter specified, the client never raises the error.
Disable SSL session reuse	If the check box is selected, the connection does not reuse the TLS session ID when reconnecting to the same AppServer or SonicMQ Adapter.

AIA Advanced Features properties

Table 63: AIA Advanced Features properties

Property	Description
Internal administrative command	Provides permission to see whether or not the AIA is running successfully. The check box is cleared by default.
Authorized IP list	If the **Internal administrative command** check box is selected, this box shows a comma-separated list of IP addresses that are allowed to access the AIA Adapter's internal administrative commands. To add or delete an IP address, click **Edit** and type or remove an IP address in the box. To allow any IP address to access the AIA Adapter's internal commands, leave the box blank. This is the default.
AppServer Keepalive	Allows the AppServer to recognize that a client bound to it is no longer connected.
Enable dynamic property updates	If the check box is selected, dynamic property updates are allowed.

Edit an AppServer Internet Adapter configuration

You can edit the configuration of an AppServer Internet Adapter.

To edit an AppServer Internet Adapter configuration:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer Internet Adapter broker instance whose configuration you want to edit.
3. Click the AppServer Internet Adapter broker instance. The **AppServer Internet Adapter** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **AppServer Internet Adapter Configuration** page appears.
5. Click **Edit**. (For details about the properties, see AppServer Internet Adapter (AIA) properties on page 128.)
6. Make the configuration changes, and then click **Save**.

Create an AppServer Internet Adapter

You can create an AppServer Internet Adapter instance.

To create an AppServer Internet Adapter instance:

1. From the drop-down for **Resources** on the management console menu, click **New OpenEdge Resource** > **AppServer Internet Adapter**.

   The **AppServer Internet Adapter Configuration** page appears.

2. Type the name of the new instance in the field provided.

   Note: The AppServer Internet Adapter name is case sensitive and can include any character except a period (.) or square brackets ([ ]). The name must be unique among all configured AppServer Internet Adapter names.

3. Click **Save**. The **AppServer Internet Adapter Configuration** page appears, showing the configuration details and allowing you to configure the instance's properties.

View the AppServer Internet Adapter log file

You can view the log file for an AppServer Internet Adapter log file by using the log file viewer. The log file viewer allows you to examine the log file through an HTML interface.

To access the log file viewer:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer Internet Adapter broker instance whose log file viewer you want to access.
3. Click the Edit icon. The **AppServer Internet Adapter** details page appears.
4. In the **Command and control** section of the page, click **Log File Viewer**.

You can work with the Log File Viewer in the following ways:

- Use the **Show** field to control how many AIA log file entries display at one time. The number entered into the **Show** field cannot be less than 10.
- Use the **Overlap** field to control how many entries are repeated from screen to screen.

  Note: The value in the **Overlap** field cannot be more than the number in the **Show** field minus one. For example, if you show 30 entries, you can overlap only 29 or fewer of them.

- Click **Reload** after changing the values in either the **Show** field or the **Overlap** field. If you do not reload, the viewer continues to display the previous values.
- Click **Go To** to control which numbered entry in the log file the viewer begins its display with. For example, a value of 10 entered into the **Go To** field will begin the display at the tenth log file entry.

  Note: You must click **Go To** after entering a value in the **Go To** field or the viewer will not update its display.

  The default display of entries is in ascending order; choose **Descending** to change the display. Note that the **Show** field dictates the number of entries shown, regardless of whether they display in ascending or descending order.

- Click **First** to display the first **x** entries, where **x** is the value in the **Show** field.
- Click **Previous** to display the previous **x** entries, where **x** is the value in the **Show** field.
- Click **Next** to display the next **x** entries, where **x** is the value in the **Show** field.

- Click **Last** to display the last **x** entries, where **x** is the value in the **Show** field.
- To view additional log file entries without changing your current starting log file entry, leave the **Go To** field blank, change the value in the **Show** field, and click **Reload**.
- If the contents of the log file have changed since you opened the viewer, the log file viewer indicates this in the **Log file status** field.
- OpenEdge Management and OpenEdge Explorer consider a viewer that has been inactive for more than four hours stale and at that point release ninety-five percent of any memory being held. If you try to use a stale viewer, OpenEdge Management and OpenEdge Explorer automatically reload the file. Because additional resource activity might have occurred during the viewer's inactivity, the reloaded log file view might not match the previous log file view of that resource.
- OpenEdge Management and OpenEdge Explorer consider a viewer that has been inactive for forty-eight hours dead. Once a viewer dies, OpenEdge Management and OpenEdge Explorer release all of its memory. To return to the log file displayed in a dead view, you must renavigate to it, even if you pinned up the view or saved a link to it before the viewer died.

Enable or disable an AppServer Internet Adapter broker

You can enable or disable an AppServer Internet Adapter broker.

To enable or disable an AppServer Internet Adapter broker:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer Internet Adapter broker instance you want to enable or disable.
3. Click the Edit icon. The **AppServer Internet Adapter** details page appears.
4. In the **Command and control** section of the page, click **Control**. The broker's **Control** page appears.
5. Click **Edit**, and either select the **Enabled** check box to enable the broker or clear the check box to disable the broker.
6. Click **Save**.

Delete an AppServer Internet Adapter broker

You can delete an AppServer Internet Adapter (AIA) instance using the management console. The instance is also deleted from the `ubroker.properties` file.

If you delete an AIA instance that is running, the instance and all associated properties are deleted from the `ubroker.properties` file. However, the AIA instance continues to run until you use the Java servlet engine to shut down the AIA.

To delete an AppServer Internet Adapter broker:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the AppServer Internet Adapter broker instance you want to delete.
3. Click the Edit icon. The **AppServer Internet Adapter** details page appears.
4. Click **Delete**, and then click **OK** to confirm the deletion.

6

Configure WebSpeed Messengers and WebSpeed Transaction Servers

You can configure property settings for WebSpeed Messengers and examine their log files in OpenEdge Management and OpenEdge Explorer.

You can configure WebSpeed Transaction Server property settings and add new WebSpeed instances in OpenEdge Management and OpenEdge Explorer. You can also start, stop, and delete WebSpeed instances, as well as view their status and log files.

The following topics provide information about WebSpeed Messenger and WebSpeed Transaction Server configuration.

For details, see the following topics:

- WebSpeed Messenger administration
- WebSpeed Transaction Server administration
- Additional WebSpeed information

WebSpeed Messenger administration

The WebSpeed Messenger resides on your Web server machine. It picks up incoming application service requests from WebSpeed clients and directs them to a WebSpeed broker that supports that application service. The Messenger is either a CGI program, or an ISAPI or NSAPI process.

There are four different WebSpeed Messengers:

- **CGIIP Messenger** — Runs on almost all Web servers, but tends to have the slowest response times.
- **WSASP Messenger** — Is used to call WebSpeed applications from a Microsoft Active Server Page. It cannot coexist with any other Messenger on your Web server.
- **WSISA Messenger** — Runs on Microsoft IIS Web servers.
- **WSNSA Messenger** — Runs on Netscape Web servers.

You cannot create or delete WebSpeed Messengers from the management console. You can, however, edit the Messenger's properties, enable or disable the Messenger, and examine the Messenger's log file from the console.

Configure a WebSpeed Messenger

You can view or edit properties in the following categories for a WebSpeed Messenger in the management console:

- General
- Controlling NameServer
- Logging Setting
- TLS
- Advanced

You cannot create or delete WebSpeed Messengers from the management console. You can, however, edit the Messenger's properties, enable or disable the Messenger, and examine the Messenger's log file from the console.

Procedures for viewing or editing the configuration are described in the topics that follow.

View or modify WebSpeed Messenger properties

You can view or modify General, Controlling NameServer, Logging Setting, TLS and Advanced properties for each WebSpeed Messenger.

To view or modify WebSpeed Messenger properties:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the WebSpeed Messenger whose properties you want to view or modify.
3. Click the WebSpeed Messenger. The **Messenger** details page appears.
4. Under **Command and control**, click **Configuration**. From this page, you can do the following:
   - View a read-only display of the Messenger properties as described in WebSpeed Messenger properties on page 139.
   - Click **Edit** to modify the broker properties. For details, see Edit a WebSpeed Messenger configuration on page 142 .

WebSpeed Messenger properties

WebSpeed Messenger General properties

Table 64: WebSpeed Messenger General properties

Property	Description
Working directory	The Messenger's working directory, where it stores temporary files and the Messenger log file
Messenger executable filename	The full path where the Messenger executable file resides

WebSpeed Messenger Controlling NameServer properties

Table 65: WebSpeed Messenger Controlling NameServer properties

Property	Description
Register with NameServer	If the check box is selected, the messenger is registered with the controlling NameServer.
Controlling NameServer	The NameServer with which the WebSpeed Messenger registers. The WebSpeed Messenger then uses the NameServer to find a WebSpeed broker that supports the current application service. (If you did not choose to register the messenger with a NameServer, the field is dimmed and unavailable.)
Minimum NameServer client port and **Maximum NameServer client port**	These properties are used to specify a range of UDP ports for the messenger instance. From this range, a messenger instance can select which UDP port to use when communicating with the NameServer. To limit the range, network administrators can specify a Minimum and a Maximum value. Specifying a range using Minimum and Maximum allows network administrators to limit the number of UDP ports needed to be open in the firewall when the NameServer client (in this case, the messenger) and the NameServer are separated by a firewall. The value for Minimum and Maximum can be zero (0) or a number between 1024 and 65535 inclusive. The Minimum value must be less than or equal to the Maximum value. If both the Minimum and Maximum values are set to zero (0), then a random port number within the implicit fixed range is dynamically assigned. How the port is assigned depends on the platform on which the messenger instance is running. If both Minimum and Maximum values are set to the same valid port number, then this UDP port number is used when communicating with the NameServer. The default value for both Minimum and Maximum is 0.

Property	Description
WebSpeed transaction server host	The WebSpeed Transaction Server host location.
WebSpeed transaction server port	The WebSpeed Transaction Server port location.

WebSpeed Messenger Logging Setting properties

For complete information about these logging settings, see *Troubleshoot ABL Applications*.

Table 66: WebSpeed Messenger Logging Setting properties

Property	Description
Log filename	Location and name of the log file.
Logging level	A value that specifies the amount of information to be written to the Messenger log. Select from the following values in the drop-down list: • **None** — Log no entries. This is equivalent to turning logging off. • **Error Only** — Log OpenEdge error messages. This includes all error messages and is unrelated to the entry types specified. Errors continue to be logged at all higher levels. • **Basic** — Logging entry types determine the logged information. Each entry type generates at least some output. This is the default. • **Verbose** — Logging entry types determine the logged information, but it is typically more information than **Basic**. • **Extended** — Logging entry types determine the logged information, but it is typically more information than **Verbose**.
Logging entry types	The one valid entry type for WebSpeed Messenger logging is MsgrTrace, which turns on logging for WebSpeed Messengers. The information logged depends on which Messenger is running and the logging level specified. For example: • **Basic** — For CGIIP, logs the start and end of the request, and the values of the CONTENT_LENGTH, PATH_INFO and QUERY_STRING variables. For other Messengers, logs the environment variables. • **Verbose** — Same as **Basic** but also includes the CGIIP environment variables and the environment variables in binary format for other Messengers. • **Extended** — Same as **Verbose**, but also includes the environment variables in binary format for CGIIP. You can specify a logging level on the entry type. For example: `MsgrTrace:3`

Property	Description
Log file threshold size	A value of 0 or a value between 500,000 and 2,147,483,647, where 0 means there is no limit on the log file size other than what the operating system imposes.
Maximum number of log files	The number of rolled-over log files to keep. The value can be 0 or a value between 2 and 999999, where 0 means there is no limit on the number of server log files to keep. The specified number represents the maximum total number of log files to keep on disk at any time, including the current log file being written to. When the file becomes equal to or greater than the **Log file threshold size**, the client process renames it and creates a new log file. The file is renamed as follows, where `######` is a number starting at 000001 and increasing to 999999, after which it rolls back over to 000001: *`filename.######.extension`*

WebSpeed Messenger TLS properties

Table 67: WebSpeed Messenger TLS properties

Property	Description
Enable SSL client connections	If the check box is selected, this property directs that the connection to the WebSpeed Transaction Server use TLS tunneling.
Disable client verification of SSL host name	If the check box is selected, this property turns off host verification for an TLS connection. If cleared, the client compares the host name for the WebSpeed Transaction Server specified by Host Name (or as returned by any Controlling NameServer) with the Common Name specified in the server digital certificate, and raises an error if they do not match. With this parameter specified, the client never raises the error.
Disable SSL session reuse	If the check box is selected, the connection does not reuse the TLS session ID when reconnecting to the same WebSpeed Transaction Server.
Disable SSL session caching	Disables session caching. (Session caching allows a client to reuse a previously established session if it reconnects prior to the session cache timeout expiring.)
SSL session cache timeout	Specifies in seconds the length of time an TLS session will be held in the session cache. The default is 180 seconds.

WebSpeed Messenger Advanced properties

Table 68: WebSpeed Messenger Advanced properties

Property	Description
Internal administration command	If the check box is selected, the Messenger allows access to the WSMAdmin Web page. The WSMAdmin Web page provides links to customize error messages, enable or disable session logging, and validate the traffic flow between the Messenger and the WebSpeed broker/agents.
Session connection ID	If the check box is selected, WebSpeed Session Connection IDs are enabled.
Authorized IP list	The list of all IP addresses that are allowed to access the WSMAdmin Web page, if it has been enabled. If you add any addresses to this list, then only requests coming from addresses that are on the list will reach the WSMAdmin Web page. This gives you the ability to tailor access to the WSMAdmin Web page, so that only users connecting from certain sites can access the administration functions available on that page.

Edit a WebSpeed Messenger configuration

You can edit the configuration of a WebSpeed Messenger.

To edit a WebSpeed Messenger configuration:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the WebSpeed Messenger whose configuration you want to edit.
3. From the list, click the WebSpeed Messenger. The **Messenger** details page appears.
4. Under **Command and control**, click **Configuration**. The messenger's **Configuration** page appears.
5. Click **Edit**, and make the changes to the properties. (For details about the properties, see WebSpeed Messenger properties on page 139.)
6. Click **Save**.

View the WebSpeed Messenger log file

You can view the log file for a WebSpeed Messenger by using the log file viewer. The log file viewer allows you to examine the log file through an HTML interface.

To access the log file viewer:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the WebSpeed Messenger whose log file viewer you want to access.

3. From the list, click the WebSpeed Messenger. The **Messenger** details page appears.
4. Under **Command and control** section, click **Log File Viewer**. You can use the Log File Viewer in the following ways:

   - Use the **Show** field to control how many database log file entries display at one time. The number entered into the **Show** field cannot be less than 10.
   - Use the **Overlap** field to control how many entries are repeated from screen to screen.

     Note: The value in the **Overlap** field cannot be more than the number in the **Show** field minus one. For example, if you show 30 entries, you can overlap only 29 or fewer of them.

   - Click **Reload** after changing the values in either the **Show** field or the **Overlap** field. If you do not reload, the viewer continues to display the previous values.
   - Click **Go To** to control which numbered entry in the log file the viewer begins its display with. For example, a value of 10 entered into the **Go To** field will begin the display at the tenth log file entry.

     Note: You must click **Go To** after entering a value in the **Go To** field or the viewer will not update its display.

     The default display of entries is in ascending order; choose **Descending** to change the display. Note that the **Show** field dictates the number of entries shown, regardless of whether they display in ascending or descending order.

   - Click **First** to display the first **x** entries, where **x** is the value in the **Show** field.
   - Click **Previous** to display the previous **x** entries, where **x** is the value in the **Show** field.
   - Click **Next** to display the next **x** entries, where **x** is the value in the **Show** field.
   - Click **Last** to display the last **x** entries, where **x** is the value in the **Show** field.
   - To view additional log file entries without changing your current starting log file entry, leave the **Go To** field blank, change the value in the **Show** field, and click **Reload**.
   - If the contents of the log file have changed since you opened the viewer, the log file viewer indicates this in the **Log file status** field.
   - OpenEdge Management and OpenEdge Explorer consider a viewer that has been inactive for more than four hours stale and at that point release ninety-five percent of any memory being held. If you try to use a stale viewer, OpenEdge Management and OpenEdge Explorer automatically reload the file. Because additional resource activity might have occurred during the viewer's inactivity, the reloaded log file view might not match the previous log file view of that resource.
   - OpenEdge Management and OpenEdge Explorer consider a viewer that has been inactive for forty-eight hours dead. Once a viewer dies, OpenEdge Management and OpenEdge Explorer release all of its memory. To return to the log file displayed in a dead view, you must renavigate to it, even if you pinned up the view or saved a link to it before the viewer died.

WebSpeed Transaction Server administration

With the WebSpeed Transaction Server, you can make WebSpeed applications available over the Web, an Intranet, or an Extranet.

You can use OpenEdge Management or OpenEdge Explorer to configure new WebSpeed Transaction Server instances and administer the configurations of existing instances. The WebSpeed Transaction Server installation provides one sample WebSpeed Transaction Server (wsbroker1). You can use this sample component as a template for creating and configuring additional WebSpeed Transaction Servers. Each WebSpeed Transaction Server is called an instance.

Use the WebSpeed Transaction Server

Each WebSpeed Transaction Server consists of a WebSpeed broker and one or more WebSpeed agents. A WebSpeed agent is a single ABL Virtual Machine (AVM) instance running on a WebSpeed Transaction Server. Each WebSpeed agent executes ABL procedures on behalf of WebSpeed clients.

To work with a WebSpeed Transaction Server instance, double-click the **WebSpeed** folder from the management console's list frame. There is one pre-defined default WebSpeed Transaction Server (wsbroker1).

The following topics describe these WebSpeed Transaction Server actions:

- Configure a WebSpeed Transaction Server on page 144
- Edit a WebSpeed Transaction Server configuration on page 163
- Create a WebSpeed broker on page 163
- Start and stop a WebSpeed broker on page 163
- Add or trim WebSpeed agents on page 164
- Delete a WebSpeed broker on page 165

Configure a WebSpeed Transaction Server

You can view or edit the following configuration properties of a WebSpeed Transaction Server broker from the management console:

- Broker
- Agent
- TLS
- Messaging
- Environment variables

You can also create a new WebSpeed Transaction Server broker, delete a broker you no longer need, or view the broker's log files.

Each of these tasks is described in the topics that follow.

View or modify WebSpeed Transaction Server broker properties

You can view of modify the following WebSpeed Transaction Server Broker properties: general properties, owner information, controlling NameServer information, AppService Name List information, logging setting details, and advanced features.

To view or modify WebSpeed Transaction Server broker properties:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the WebSpeed Transaction Server broker whose properties you want to view or modify.
3. Click the WebSpeed Transaction Server broker. The **WebSpeed** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **WebSpeed Configuration** page opens with that Broker's properties in focus. From this page, you can:
   - View a read-only display of the WebSpeed broker configuration properties as described in WebSpeed Transaction Server broker properties on page 145. All these tables follow this procedure.
   - Click **Edit** to modify the broker properties. For details, see Edit a WebSpeed Transaction Server configuration on page 163.

WebSpeed Transaction Server broker properties

The following topics describe the WebSpeed Transaction Server broker properties.

WebSpeed Broker General properties

Table 69: WebSpeed Broker General properties

Property	Description
Auto start	Indicates whether or not the server automatically starts when the controlling AdminServer starts.
Port number	The number of the TCP/IP port that the server broker listens on.
Working directory	Your working directory. The default is the working directory set during OpenEdge installation.
Operating Mode	The operating mode for this server.

WebSpeed Broker Owner Information properties

Table 70: WebSpeed Broker Owner Information properties

Property	Description
Group name	(UNIX only) The group name on UNIX.
Username	The username of an account that has system-administrative rights.
Password	The username account password.
Password Confirm	Confirmation of the username account password. This field appears only if you are editing the configuration. Note that this field does not appear if the AdminServer is running on a non-Windows host machine.

WebSpeed Broker Controlling NameServer properties

Table 71: WebSpeed Broker Controlling NameServer properties

Property	Description
Register with NameServer	Indicates whether or not to register the broker with a controlling NameServer.
Controlling NameServer	The name of the NameServer with which this broker registers. (If you did not select to register the broker with a NameServer, the field is unavailable.)
Registration mode	How the broker specifies its hostname if it is registering with a controlling NameServer. This hostname information is passed onto a client application when it attempts to connect to an application service that the broker supports. The choices are: • **Register-IP** — (Default) This setting is the most efficient mechanism, and can be used in most cases. It registers with the IP address of the machine where the broker is located. • **Register-LocalHost** — The broker registers with the hostname of the machine that it runs on. Use this setting when the broker runs on a machine with a single hostname and more than one IP address. • **Register-HostName** — The broker registers with the values specified in the HostName property. Use this setting when your clients need a fully qualified host name to connect to a broker in a different DNS domain.
Registration host name	The name of the non-local host.

WebSpeed Broker AppService Name List properties

Table 72: WebSpeed Broker AppService Name List properties

Property	Description
Application service names	A list to which you can add or delete application service names.
Supports default service	Indicates whether the AppServer broker supports the default service.

WebSpeed Broker Logging Setting properties

Table 73: WebSpeed Broker Logging Setting properties

Property	Description
Broker log filename	The broker log filename.

Property	Description
Broker logging level (dynamic property)	A value that specifies the amount of information that is written to the broker log. The following values are available: • **None** — Log no entries. This is equivalent to turning logging off. • **Error Only** — Log OpenEdge error messages. This includes all error messages and is unrelated to the entry types specified. Errors continue to be logged at all higher levels. • **Basic** — Broker logging entry types determine the logged information. Each entry type generates at least some output. This is the default. • **Verbose** — Logging entry types determine the logged information, but it is typically more information than **Basic**. • **Extended** — Logging entry types determine the logged information, but it is typically more information than **Verbose**. You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.
Append to broker log file	Indicates whether or not a new broker log file should be created when the broker is started.

Broker logging entry types (dynamic property)	A comma-separated list of the following valid entry types for agent logging:
	• **UBroker.AutoTrim** — Logs messages about automatic agent thread trimming by the broker based on the Auto trim timeout property. Information is logged at **Basic** and higher.
	• **UBroker.Basic** — Logs messages about broker and agent startup and shutdown at the following logging levels:
	  • **Basic** — Logs NameServer registration and connections from clients.
	  • **Verbose** — Logs information about all the property values read from the `ubroker.properties` file for that broker instance, and more information on the connection from clients.
	• **UBroker.ThreadPool** — Logs messages about the pool of threads managed by the broker. These threads are used to control the client requests and the agent processes. This entry type logs messages when adding, removing, and communicating with these threads. Information is logged at **Basic** and higher.
	You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.
	Turn on the following logging entry types only at the request of Progress Software Corporation Technical Support:

Property	Description
	• **UBroker.ClientFSM** — Logs messages about the state of the client thread as it processes the requests, from the moment it receives a connection request to the end of the request. Information is logged at **Basic** and higher. • **UBroker.ServerFSM** — Logs messages about the state of the agent thread as it processes the requests, from the moment it receives a connection request to the end of the request. Information is logged at **Basic** and higher. • **UBroker.ClientMsgStream** — Logs messages about messages exchanged between the broker and client threads in binary format. Information is logged at **Basic** and higher. • **UBroker.ServerMsgStream** — Logs messages about messages exchanged between the broker and agent threads in binary format. Information is logged at **Basic** and higher. • **UBroker.ClientMsgQueue** — Logs messages about the client thread as it processes queued messages. Information is logged at **Basic** and higher. • **UBroker.ServerMsgQueue** — Logs messages about the agent thread as it processes queued message. Information is logged at **Basic** and higher. • **UBroker.ClientMemTrace** — Traces messages being processed by the client threads. Information is logged at **Basic** and higher. • **UBroker.ServerMemTrace** — Traces messages being processed by the agent threads. Information is logged at **Basic** and higher. • **UBroker.Stats** — Logs statistics about the number of requests, plus maximum, minimum, and average request wait and duration times. Information is logged at **Basic** and higher. • **UBroker.All** — Logs all UBroker entry types. You can specify a separate logging level for each log entry type and use a wildcard (*) to specify multiple entry types. For example: `UBroker.Basic:3,UBroker.ThreadPool:4, UBroker.Client*`

Property	Description
Broker log file threshold size	A value of 0 or a value between 500,000 and 2,147,483,647, where 0 means there is no limit on the log file size other than what the operating system imposes.
Maximum number of broker log files	The number of rolled-over log files to keep. The value can be 0 or a value between 2 and 999999, where 0 means there is no limit on the number of broker log files to keep. The specified number represents the maximum total number of log files to keep on disk at any time, including the current log file being written to. When the file becomes equal to or greater than the **Broker log file threshold size**, the client process renames it and creates a new log file. The file is renamed as follows, where ###### is a number starting at 000001 and increasing to 999999, after which it rolls back over to 000001: *filename.######.extension*

WebSpeed Broker Advanced Features properties

Table 74: WebSpeed Broker Advanced Features properties

Property	Description
Maximum client instances	The maximum number of client connections that the broker can support concurrently. The default is 512, which is high enough to ensure that the number of client connections is virtually unlimited. Realistically, however, the system-level resources required to support an unlimited number of clients might be exhausted before this limit is reached. If you see broker log entries that contain Exception Messages for "out of space" or "OutOfMemory" you might need to reduce the maximum number of client connections to a more reasonable value. Note, however, that lowering this value can cause some client requests to be rejected ("Exceeded Max Clients"). Configuring and starting multiple brokers to handle higher client loads will alleviate the problem.
Priority weight (0-100) (dynamic property)	An integer value between 0 and 100 that influences the share of the workload that the selected AppServer receives. The larger the value, the heavier the load that is distributed to the server. The NameServer distributes client requests across all WebSpeed Transaction Servers that have the same application service in proportion to the **Priority weight** value. The default is 0. You can update this property dynamically. Any changes will affect all current and new brokers.

Property	Description
Registration retry (in seconds)	After registering with its controlling NameServer, the WebSpeed Transaction Server periodically sends "keep-alive" messages to the NameServer to let the NameServer know that it is still active. The **Registration retry** value is the number of seconds that pass between "keep-alive" messages. The default is 30.
Server startup timeout (dynamic property)	Specifies the amount of time, in seconds, that the broker waits for an active agent to become available before starting a new agent process. The default is 3. You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.
Request timeout (dynamic property)	Specifies the amount of time, in seconds, that the broker waits for an agent to become available for processing a request. The broker waits for this period only if the **Maximum servers** setting has been reached. After the timeout has expired, the client receives a "no servers available" error message. The default is 15. You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.
Auto trim timeout (dynamic property)	The amount of time, in seconds, that the broker waits before automatically trimming the number of running agents. The broker keeps track of the maximum number of agents that are simultaneously busy during the interval you specify. At the end of the interval, the broker attempts to trim the number of agents to match either the **Maximum servers** for the interval, or the **Minimum servers**, whichever is greater. You can disable this feature by setting the timeout to zero (0). The default **Server Auto-trim Interval** is 1800 seconds. Progress Software recommends that you use this default setting to avoid unnecessary process management that might result from frequent starting and trimming of agents. You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.

Property	Description
TCP/IP version	Internet Protocol for network communication: • **IPv4** — Accepts only IPv4 connections • **IPv6** — Allows IPv4 and IPv6 connections using mapped address (where supported) The default is IPv4. You must also set the **jvmArgs** property in the `ubroker.properties` file to override the default behavior of the JVM for IPv6 connections. `jvmArgs=-Djava.net.preferIPv4Stack=false -Djava.net.preferIPv6Addresses=tru`
jvmargs	Java System Properties.
Enable dynamic property updates	Allows dynamic changes to occur to several of the instance's properties, without requiring you to shut down the instance. This property is disabled by default. **Note:** Any property that is dynamic is labeled as such in this document.
Enable debugging through the broker (dynamic property)	Specifies whether or not the debugger is enabled; select the check box to enable it. If you disable this property while the debugger is connected, the debugger connection terminates. **Note:** The **Broker debugger port number** property must also be set to a valid value for the debugger to be properly enabled. This property is disabled by default.
Broker debugger port number (dynamic property)	Specifies the port number that the broker debugger will use when it starts. You can choose any valid value; if the port you specify is already in use, the debugger service will fail to start. If you change the port number while the debugger service is active, the debugger service will be shut down and restarted using the new port number. The default port number is 3099.

Property	Description
Enable SSL for broker debugger (dynamic property)	Specifies whether the broker debugger should accept only TLS-encrypted connections. If you enable this property, the Progress Developer Studio debugger will be required to establish an TLS session. If you enable or disable the property while the debugger service is active, the debugger will be disconnected. The debugger service will be shut down and then restarted using the new selection. This option is disabled by default.
Use the broker's private key/digital certificate for debugger (dynamic property)	When enabled, indicates that, when using TLS, the broker debugger service should use the same private key as used by the AppServer agents. If this property is enabled, then the **Debugger private key/digital certificate alias name** and **Password to access the debugger's key/certificate** property settings are ignored. This property is enabled by default.
Debugger private key/digital certificate alias name (dynamic property)	The name of the private key that should be used for the debugger service when the agent's private key is not being used. This property must be defined if **Enable SSL for broker debugger** is enabled.
Password to access the debugger's key/certificate (dynamic property)	The password required to access the private key specified by **Debugger private key/digital certificate alias name**. The default for this property is blank.
Broker debugger passphrase (dynamic property)	An optional property that represents the passphrase required by the debugger to connect to the debugger service in the broker. If the debugger is connected to the debugger service when the passphrase is changed, the debugger remains connected. The next time the debugger attempts to connect to the debugger service, the new password is required. The default for this property is blank.
Broker debugger passphrase Confirm	Confirms the passphrase provided in **Broker debugger passphrase**.
Publish directory	The name of the local directory where you want r-code to be placed when you are publishing from Progress Developer Studio. If you are publishing remotely from Progress Developer Studio, you must set this property.

View or modify WebSpeed Agent properties

You can set the following agent properties for a WebSpeed Broker: general properties, logging setting details, pool range, and advanced features.

To view or modify WebSpeed Agent properties:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the WebSpeed Transaction Server broker whose agent properties you want to view or modify.
3. Click the WebSpeed Transaction Server broker. The **WebSpeed** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **WebSpeed Configuration** page opens with the **Broker** properties in focus.
5. Click the **Agent** tab. From this page, you can do the following:
   - View a read-only display of the WebSpeed Agent properties as described in WebSpeed Agent properties on page 153.
   - Click **Edit** to modify the broker properties. For details, see Edit a WebSpeed Transaction Server configuration on page 163.

WebSpeed Agent properties

The following topics describe the WebSpeed Agent properties.

WebSpeed Agent General properties

Table 75: WebSpeed Agent General properties

Property	Description
Server executable file (dynamic property)	Either the default agent executable pathname or a different agent executable file of your choosing. You generally only need to specify a nondefault value if you have generated a new executable using the OEBUILD utility. You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.
Server startup parameters (dynamic property)	The OpenEdge startup parameters that you want to specify to start each agent. You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.
PROPATH	The search path that agents use to locate ABL procedures that they execute. Specify a list of directory pathnames in the form of an OpenEdge PROPATH. Make sure that you have copied all of your WebSpeed Transaction Server procedures to one or more of the directories in the PROPATH.

Property	Description
Minimum port number (dynamic property)	The minimum TCP/IP port number from a range that each agent process can listen on. When each agent starts, it allocates a port that is not being used from the specified range, up to the **Maximum port number** setting. You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.
Maximum port number (dynamic property)	The maximum TCP/IP port number from a range that each agent can listen on. When each agent starts, it allocates a port that is not being used from the specified range, down to the **Minimum port number** setting. You can update this property dynamically. Any changes will affect only new agents that start after the property value has changed; existing (running) brokers and agents are not updated.
Server application mode	Development or Production.
Flush statistical data (dynamic property)	How often the data is flushed by the agent to the broker. This is expressed as the number of remote procedure calls between flushes. You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.

WebSpeed Agent Logging Setting properties

Table 76: WebSpeed Agent Logging Setting properties

Property	Description
Server log filename	The server log file name.

Property	Description
Server logging level (dynamic property)	A value that specifies the amount of information to be written to the agent log. You can select from the following values in the drop-down list: • **None** — Log no entries. This is equivalent to turning logging off. • **Error Only** — Log OpenEdge error messages. This includes all error messages and is unrelated to the entry types specified. Errors continue to be logged at all higher levels. • **Basic** — Server logging entry types determine the logged information. Each entry type generates at least some output. This is the default. • **Verbose** — Logging entry types determine the logged information, but it is typically more information than **Basic**. • **Extended** — Server logging entry types determine the logged information, but it is typically more information than **Verbose**. You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.
Append to server log file	Indicates whether a new agent log file should be created when the WebSpeed Transaction Server is started.

Server logging entry types (dynamic property)	A comma-separated list of valid entry types for agent logging:
	• **4GLTrace** — Turns on the logging for the execution of the following ABL statements: RUN, FUNCTION (user-defined functions), PUBLISH and SUBSCRIBE. Logs information at **Basic** or higher.
	• **ASDefault** — Combines the **ASPlumbing** and **DB.Connects** log entry types. This is the default value for AppServer agents.
	• **ASPlumbing** — Turns on logging for different actions depending on the logging level specified:
	  • **Basic** — Logs messages when an AppServer agent starts and stops, and when client processes connect and disconnect from the AppServer agent.
	  • **Verbose** — Same as **Basic**, plus logging of the execution of the top level procedure (that is, the procedure that the client process asks the AppServer agent to execute on its behalf, but not any procedure that the top level procedure executes).
	  • **Extended** — Same as **Verbose**, plus messages about agent state, and status messages about read and write socket operations. Do not turn this on unless instructed to do so by Progress Software Technical Support.

Property	Description

	• **DB.Connects** — Turns on logging of database connections and disconnections. The messages include database name and user ID number. Logs information at **Basic** or higher.
	• **DynObjects.DB** — Turns on logging for dynamic database-related objects (for example, TEMP-TABLE or DATASET) as they are created and destroyed. Logs information at **Basic** or **Verbose**.
	• **DynObjects.XML** — Turns on logging for dynamic XML-related objects (for example, x-document or x-noderef) as they are created and destroyed. Logs information at **Basic** or **Verbose**.
	• **DynObjects.Other** — Turns on logging for dynamic objects that do not match the other dynamic object categories (for example, procedure or server socket) as they are created and destroyed. Logs information at **Basic** or **Verbose**.
	• **ProEvents.UI.Char** — Turns on logging of keystroke events for printable characters. Logs information at **Basic** or higher.
	• **ProEvents.UI.Command** — Turns on logging of keystroke events for nonprintable characters at the following logging levels:
	  • **Basic** — Logs nonprintable keystrokes and a subset of Windows GUI events
	  • **Verbose** — Logs nonprintable keystrokes and all Windows GUI events
	• **ProEvents.Other** — Turns on logging of COM, asynchronous, and server socket events. Logs information at **Basic** or higher, beginning with events for which you have written triggers.
	• **QryInfo** — Turns on database query information logging at the following logging levels, with the same amount of information logged for each level:
	  • **Basic** — Logs static queries and initial dynamic queries
	  • **Verbose** — Logs static queries and multiple instances of dynamic queries
	• **SAX** — Turns on logging for various stages of execution using the ABL SAX parser.
	You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.You can specify a separate logging level for each log entry type, and you can use a wildcard (*) to specify multiple entry types. For example:
	`4GLTrace:3,ASPlubing:2,DynObjects.*`

Property	Description
Server log file threshold size	A value of 0 or a value between 500,000 and 2,147,483,647, where 0 means there is no limit on the agent log file size other than what the operating system imposes.
Maximum number of server log files	The number of rolled-over log files to keep. The value can be 0 or a value between 2 and 999999, where 0 means there is no limit on the number of agent log files to keep. The specified number represents the maximum total number of log files to keep on disk at any time, including the current log file being written to. When the file becomes equal to or greater than the **Server log file threshold size**, the client process renames it and creates a new log file. The file is renamed as follows, where ###### is a number starting at 000001 and increasing to 999999, after which it rolls back over to 000001: *filename.######.extension*
Server watch dog interval	The interval (in seconds) for the server log file watchdog thread. If the **Server log file threshold size** property is specified, the broker will roll over the server log file when its size is larger than the logThreshold value.

WebSpeed Agent Pool Range properties

Table 77: WebSpeed Agent Pool Range properties

Property	Description
Initial number of servers to start	The number of agents you want the broker to start initially.
Minimum servers	The minimum number of agents before the broker starts additional agents. If you trim the number of agents below this value, the server starts any additional agents needed to maintain the specified minimum when the next client connects.
Maximum servers	The maximum number of agents that this server can have running at the same time.

WebSpeed Agent Advanced Features properties

Table 78: WebSpeed Agent Advanced Features properties

Property	Description
Application URL	A server-relative URL (starting with a "\") to the path of the application. If the Web server redirects an incoming URL such that the SCRIPT_NAME and PATH_INFO variables do not refer to the application, setting this property allows an application to create self-referencing URLs when SCRIPT_NAME and/or PATH_INFO are not correct. If this option is set, the value of the applicationURL global variable available to applications is set to this value instead of a combination of SCRIPT_NAME and PATH_INFO.
Default cookie path	A server-relative URL (starting with a "\") to use for all cookies if not explicitly specified in the set-cookie( ) or delete-cookie( ) functions. The default is the value of the AppURL global variable.
Default cookie domain	Domain to send cookies. The default is blank, causing the Web browser to set cookies only for the current host. If cookies must be available to multiple hosts in the same domain by default, set that domain name here; for instance, `.progress.com`. Unless a value is specified for the domain option for set-cookie( ) and delete-cookie( ) functions, this value is used.
File upload directory	Specifies a directory where WebSpeed servers can upload files. Refer to the `ubroker.properties` file for an HTML snippet that shows how to use this feature. By default this capability is disabled. To enable it, specify an upload directory. Only text file upload is supported, not binary.
Binary upload max size	Specifies the maximum size (in bytes) of binary files that WebSpeed handles. The application can access the contents of the binary files through the ABL language. Note that this property is not related to the **File upload directory** property; therefore, WebSpeed will not upload the file to a directory. You can set any of the following options: • **0** — Binary upload is disabled. • **-1** — No maximum size of a binary upload; binary upload enabled. You can alternatively type any non-zero value greater than 0 as the maximum size of the binary upload.
Debug mode	Whether the debug mode is enabled, disabled, or using the default.

Property	Description
4GL debugger enabled	Whether the debugger is enabled or disabled: • **Disabled** — Specifies ?debug=on in the URL has no effect • **Enabled** — Allows debug information if ?debug=on is in the URL
Execution Time Limit	Sets the maximum time in seconds that a remote procedure can execute on a given Web server. The time limit applies to all remote procedures that execute on the Web server. The default value is zero, which indicates that no time limit exists.

View or modify WebSpeed TLS properties

You can set properties in the following WebSpeed Broker TLS categories: general properties and advanced features.

To view or modify WebSpeed TLS properties:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the WebSpeed Transaction Server broker whose TLS properties you want to view or modify.
3. Click the WebSpeed Transaction Server broker. The **WebSpeed** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **WebSpeed Configuration** page opens with the **Broker** properties in focus.
5. Click the **SSL** tab. From this page, you can:
   - View a read-only display of the WebSpeed TLS properties as described in WebSpeed TLS properties on page 159. Both of these tables follow this procedure.
   - Click **Edit** to modify the TLS properties. For details, see Edit a WebSpeed Transaction Server configuration on page 163.

WebSpeed TLS properties

The following topics describe the WebSpeed TLS properties.

WebSpeed TLS General properties

Table 79: WebSpeed TLS General properties

Property	Description
Enable SSL client connections	If the check box is selected, this property specifies that all connections to this WebSpeed Transaction Server must use TLS tunneling.

Property	Description
Private key/Digital certificate alias name	The alias name within the OpenEdge keystore of the private key and digital certificate entry to use to authenticate all connections to this WebSpeed Transaction Server. If not otherwise selected, the WebSpeed Transaction Server uses the default_server server certificate alias.
Password to access the key/certificate	The password to use for accessing the private key and digital certificate. You must specify a password when you specify the select an OpenEdge keystore alias name. **Note:** Your password is encrypted in the `ubroker.properties` file. If you set the password in `ubroker.properties` manually, you must specify the password as an encrypted value. You can obtain this value using the genpassword utility located in the bin directory of your OpenEdge installation. If you use the default_server server certificate, it also has a default password that you do not need to specify.
Password to access the key/certificate Confirm	Confirmation of the username account password. This field appears only if you are editing the configuration. Note that this field does not appear if the AdminServer is running on a non-Windows host machine.

WebSpeed TLS Advanced Features properties

Table 80: WebSpeed TLS Advanced Features properties

Property	Description
Disable SSL session caching	If this property is checked, caching for the TLS client session is disabled.
SSL session cache timeout	The length of time, in seconds, that an TLS client session is held in the session cache, during which an TLS client can resume its session. The default is 180.

View or modify WebSpeed Messaging properties

You can view or modify several WebSpeed Messaging properties.

To view or modify WebSpeed Messaging properties:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the WebSpeed Transaction Server broker whose messaging properties you want to view or modify.

3. Click the WebSpeed Transaction Server broker. The **WebSpeed** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **WebSpeed Configuration** page opens with the **Broker** properties in focus.
5. Click the **Messaging** tab. From this page, you can:
   - View a read-only display of the WebSpeed Messaging properties as described in , which follows this procedure.
   - Click **Edit** to modify the WebSpeed Messaging properties. For details, see Edit a WebSpeed Transaction Server configuration on page 163.

WebSpeed Messaging properties

Table 81: WebSpeed Messaging properties

Property	Description
SonicMQ ServerConnect enabled	Allows the asbroker or wsbroker agent to start a personal SonicMQ Adapter instead of going to a separate one running elsewhere.
SonicMQ ServerConnect broker log filename	A valid path and file name.
Broker logging level (dynamic property)	A value that specifies the amount of information to be written to the SonicMQ ServerConnect broker log. You can select from the following values in the drop-down list: • **None** — Log no entries. This is equivalent to turning logging off. • **Error Only** — Log error messages. This includes all error messages and is unrelated to the entry types specified. Errors continue to be logged at all higher levels. • **Basic** — Broker logging entry types determine the logged information. Each entry type generates at least some output. This is the default. • **Verbose** — Broker logging entry types determine the logged information, but it is typically more information than **Basic**. • **Extended** — Broker logging entry types determine the logged information, but it is typically more information than **Verbose**. You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.
Append to broker log file	Indicates whether a new broker log file should be created when the SonicMQ ServerConnect broker is started, even if the broker log file already exists.
SonicMQ ServerConnect server log filename	A valid path and file name.

Property	Description
Server logging level (dynamic property)	A value that specifies the amount of information to be written to the SonicMQ ServerConnect server log. Each logging level name has the indicated numeric value. You can select from the following values in the drop-down list: • **None** — Log no entries. This is equivalent to turning logging off. • **Error Only** — Log error messages. This includes all error messages and is unrelated to the entry types specified. Errors continue to be logged at all higher levels. • **Basic** — Broker logging entry types determine the logged information. Each entry type generates at least some output. This is the default. • **Verbose** — Broker logging entry types determine the logged information, but it is typically more information than **Basic**. • **Extended** —Broker logging entry types determine the logged information, but it is typically more information than **Verbose**. You can update this property dynamically. Any changes will affect all current and new brokers and/or agents.
Append to server log file	Indicates whether a new server log file should be created, even if the broker log file already exists, when the SonicMQ ServerConnect instance is started.

Set or delete WebSpeed environment variables

You can set or delete environment variables. For WebSpeed, environment variables that appear in the list are scoped to:

- The process in which the WebSpeed Transaction Server broker executes
- Every WebSpeed Transaction Server agent that the WebSpeed Transaction Server broker starts

To set or delete WebSpeed environment variables:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the WebSpeed Transaction Server broker whose environmental variables you want to set.
3. Click the WebSpeed Transaction Server broker. The **WebSpeed** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The broker's **Configuration** page appears.
5. Click **Edit**.
6. Click **Environment Variables**.
7. Enter each variable name and then either provide its value in the format `name=value`, or remove an existing variable.
8. Click **Save** when you finish adding (or deleting) the variables and their values.

Edit a WebSpeed Transaction Server configuration

You can edit a WebSpeed Transaction Server configuration.

To edit a WebSpeed Transaction Server configuration:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the WebSpeed Transaction Server broker whose configuration you want to edit.
3. Click the WebSpeed Transaction Server broker. The **WebSpeed** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **WebSpeed Configuration** page opens.
5. Click **Edit**. Make the changes you want to the properties. (For details about the properties, see WebSpeed Messenger properties on page 139, WebSpeed Transaction Server broker properties on page 145, WebSpeed Agent properties on page 153, WebSpeed TLS properties on page 159, WebSpeed Messaging properties on page 161.)
6. Click **Save**. Any changes you make are also saved in the `ubroker.properties` file.

Create a WebSpeed broker

OpenEdge Management and OpenEdge Explorer provide a sample WebSpeed broker instance (wsbroker1) with a default configuration. Each new instance you create uses the same default configuration, which you can edit.

To create a WebSpeed broker instance:

1. From the drop-down for **Resources** on the management console menu, click **New OpenEdge Resource** > **WebSpeed**. The **WebSpeed Configuration** page appears.
2. Type the name of the new WebSpeed broker instance in the field provided.

   Note: The WebSpeed broker instance name is case sensitive and can include any character except a period (.) or square brackets ([ ]). The name must be unique among all configured WebSpeed broker instance names.

3. Click **Save**. The **WebSpeed Configuration** page appears, allowing you to configure the WebSpeed broker's properties.

Each new WebSpeed broker instance you create uses the default configuration. However, several properties, such as port numbers, must have unique values for each WebSpeed broker in order for the instance to operate properly.

Start and stop a WebSpeed broker

From the management console, you can start either a local WebSpeed broker or a WebSpeed broker running under a remote AdminServer on another machine. You must first connect to the AdminServer that manages the WebSpeed broker you want to start.

You can also choose to start a WebSpeed broker automatically when the AdminServer starts.

Note: Prior to starting a WebSpeed broker using SonicMQ ServerConnect, you must first configure the WebSpeed broker as SonicMQ ServerConnect enabled.

To start or stop a WebSpeed broker, refer to the *Starting or Stopping OpenEdge resources* section in *OpenEdge Management: Servers, DataServers, Messengers, and Adapters*.

Start a WebSpeed broker automatically

You can set a WebSpeed broker to start automatically when the AdminServer starts.

To start a WebSpeed broker automatically:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the WebSpeed Transaction Server broker you want to start automatically.
3. Click the Edit icon. The **WebSpeed** details page appears.
4. In the **Command and control** section of the page, click **Configuration**. The **WebSpeed Configuration** page appears.
5. Click **Edit**.
6. Select the **Auto start** option by clicking in the check box.
7. Click **Save**.

Add or trim WebSpeed agents

Using OpenEdge Management or OpenEdge Explorer, you can start additional WebSpeed agents, up to the number specified in the Maximum Agents property setting.

You can also trim running WebSpeed agents, down to the Minimum Agents property setting.

More information about trimming WebSpeed Agents

To better understand the use of the Agent Auto-trim feature, consider a scenario involving a broker serving a client/server application running in stateless mode. Assume that the following property values are set:

- Minimum Agent/Server Instances: 10
- Initial Agent/Server Instances to Start: 10
- Maximum Agent/Server Instances: 50
- Agent/Server Auto-trim Interval: 1800

When the broker starts, it starts 10 agents. This also marks the start of the first Agent/Server Auto-trim Interval. Assume that the broker is idle for 30 minutes (it is just before people arrive for work, for instance). At the end of the 30-minute interval, the highest number of agents that were busy at the same time would be zero. (Since there were no requests made of any of the agents, none of the agents was busy.) However, since zero is less than the Minimum Agent/Servers Instances value of 10, no automatic trimming takes place. This marks the start of the second interval.

As the second interval starts, employees begin their workday. There is a sudden demand for agents; in fact, the broker receives so many requests that it must start 20 additional agents. Assume that the high volume of workload, and demand on the agents, continues through the second interval. At the interval's end, the maximum number of busy agents is 30, which is the number the broker scaled up to in response to high demand. Thirty agents are running, and the maximum busy level is 30. No trimming occurs.

During the next interval, the request load diminishes somewhat. At the end of the interval, only 20 agents are concurrently busy. The maximum number of busy agents for the interval remains at 30, since 30 were busy at the start of the interval. Thirty agents are running, and the maximum busy level is 30. No trimming occurs.

Assume that all employees have to attend a company-wide meeting that begins midway through the interval. The highest number of agents busy at that same time was 20. At the end of the interval, there are 30 agents running, with zero agents currently busy. Since the maximum busy level is 20 (at the start of the interval), and the number of running agents is 30 (all are available since everyone is at the company meeting), the broker attempts to trim back the number of agents to 20.

Assume that the company meeting ran for another half-hour, and the entire next interval elapsed without any requests. The maximum busy agents is zero, with 20 currently running. Since the Minimum Agent/Server Instances property value is set to 10, the broker attempts to trim back to 10 agents.

Keep in mind that the lower you set the Agent/Server Auto-trim Interval value, the more quickly the broker reacts to a drop in the workload. Under a reasonably steady work load, the broker should trim the number of running agents to the number needed. However, if the workload fluctuates a great deal and the Agent/Server Auto-trim Interval is set too low, the broker may trim agents that it must then restart. As stated earlier, repetitive trimming and restarting causes the system to waste resources on unnecessary process management.

Add or trim WebSpeed agents

To add or trim WebSpeed agents:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the WebSpeed Transaction Server broker to which you want to add or trim agents.
3. Click the WebSpeed Transaction Server broker. The **WebSpeed** details page appears.
4. If the WebSpeed broker is not started, click **Broker Control**. Then click **Start WebSpeed**.
5. When the WebSpeed broker is running, return to the **Details** page.
6. In the **Command and control** section of the page, click **Agent Pool Control**.
7. Review the **Agent pool initial configuration**, the **Agents state**, and the **Agents pool summary** information.
8. To add or trim agents, choose the corresponding option in the **Add**/**Trim** dropdown menu, and type the number of agents in the box provided.
9. Click **Submit**.

Delete a WebSpeed broker

You can delete any WebSpeed broker instance listed in the management console as long as the instance is not running. If the instance is running, you must stop it and then delete it.

To delete a WebSpeed broker instance:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter, or search for, the WebSpeed Transaction Server broker instance you want to delete.

3. Click the WebSpeed Transaction Server broker instance. The **WebSpeed** details page appears.
4. Click **Delete**, and then click **OK** to confirm the deletion.

   The **OpenEdge Management Resources** page appears. A set of next-step options that are related to resources are available for your use.

Additional WebSpeed information

You can find additional information about using OpenEdge Management and/or OpenEdge Explorer with WebSpeed in *OpenEdge Management: Servers, DataServers, Messengers, and Adapters*. See that document for details, including specifics about the following:

- Working with the WebSpeed log file monitor and viewer
- Killing a WebSpeed process

7

Configure SonicMQ Adapters

You can work with SonicMQ Adapter instances in OpenEdge Management and OpenEdge Explorer.

Support for SonicMQ Adapters has been deprecated from OpenEdge Managmenet from this release. However, you can configure a Remote AdminServer to continue to manage and monitor SonicMQ Adapter for 11.7x and 12.1. See, Set up a Remote AdminServer.

8

Configure OE Web Servers

You can work with OE Web Server instances in OpenEdge Management and OpenEdge Explorer, as described in the following topics.

For details, see the following topics:

- Configure and manage OE Web Servers
- Manage OE Web Servers
- Work with REST Web Applications

Configure and manage OE Web Servers

You can configure and manage OE Web Servers by:

- REST Management Agent on page 170
- OE Web Server on page 170
- REST Web Applications on page 170
- Apache Tomcat on page 170
- Install a REST Management Agent into a Java AdminServer on page 170
- Manage REST Web Applications with the OE Web Server on page 171

Each of these definitions and tasks are described in the following topics.

REST Management Agent

A REST Management Agent is a component of Progress OpenEdge that you use to deploy, configure, and manage a Java AdminServer's OpenEdge REST Web Applications. The REST Management Agent is a Java Web application that acts as an intermediary for you to communicate with the deployed REST Web Applications. It helps you enable/disable the application, collect statistics, and provide a run-time configuration for the REST Web Applications.

OpenEdge ships with a default REST Management Agent (oerm.war).

OE Web Server

An OE Web Server is a remote Web Server application that manages the deployment and management of REST Web Applications.

OpenEdge provides three clients—OpenEdge Management, OpenEdge Explorer, and RESTMAN utility—that use an OE Web Server to manage REST Web Applications.

REST Web Applications

A REST Web Application is a Web application archive file (WAR file) that includes one or more REST services. A REST service is a service defined to expose your REST service annotated ABL applications to a Web Server as a REST Web application.

A REST Web application typically includes java classes, web pages (HTML), images, and other files that together make up a web application.

Apache Tomcat

The default Web Server shipped with OpenEdge is where you can deploy the REST Web Applications that you want to make available to clients. The Tomcat server is available in your OpenEdge installation directory, *OpenEdge-install-dir*/servers/tomcat.

Install a REST Management Agent into a Java AdminServer

As part of the OpenEdge installation, REST Management Agent is installed in the OpenEdge installation directory, *OpenEdge-install-dir*\servlets\rest.

Note: You need not perform the following steps if you are using the Java AdminServer installed with the Apache Tomcat shipped with OpenEdge installation as Tomcat already has the OE Web Server installed.

To start working with OE Web Server and REST Web Applications, you must configure your Web Server to install the REST Management Agent in the Java AdminServer associated with the Web Server. To configure your Web Server:

Copy the REST Management Agent, `oerm.war`, from the directory, *OpenEdge-install-dir/servlets/rest*, to the Web Server deployment directory.

For example, if you are using a Tomcat Web Server, you copy `oerm.war` to the *$CATALINA_HOME/webapps* directory.

Note: If you are using Tomcat Web Server you can also deploy `oerm.war` using the Tomcat manager. For more information, see the Tomcat manager documentation.

Note: If the Web Server is already running, you may need to restart the Web Server for the new configuration settings to take effect. New applications are normally picked up automatically by the Tomcat Web Server.

You can query the OE Web Server Web application by browsing to the URL `http://host-name:port-number/oerm/index.html` to check whether the REST Management Agent has been correctly installed and configured. For instance, if you are using the Tomcat server and you are using the default ports, you can browse the URL `http://localhost:8080/oerm/index.html`.

Manage REST Web Applications with the OE Web Server

The OE Web Server is installed in a Web Server as a single Web application that you can instantiate as one or more OE Web Server instances. Each OE Web Server instance provides a single point of management for one or more deployed REST Web Applications. Thus, to begin working with the OE Web Server in OpenEdge Management or OpenEdge Explorer:

1. Create a new OE Web Server instance if none exists or you need another, and configure it for operation. (The OE Web Server comes installed with one local OE Web Server instance (restmgr1)).
2. For an OE Web Server instance where you want to deploy a new REST Web application, optionally review and update the default property settings that govern the run-time behavior of new REST Web application deployments. You can also adjust these settings for each REST Web application individually after you deploy it.
3. Deploy the new REST Web application into the context of an OE Web Server instance that you have configured.
4. As you test the REST Web application, review the OE Web Server instance status and statistics, and update any OE Web Server instance properties as needed. Such changes affect all REST Web Applications deployed to the same OE Web Server instance.

   Note: To apply changes to most REST Web application properties, you must restart the Web Server where the REST Web application is installed. You can make temporary changes to selected REST Web application properties at run time without restarting the Web Server. However, you must update the REST Web application instance configuration to make these changes permanent.

5. As you client-test the REST Web application, review the REST Web application status and statistics, and update any REST Web application properties as needed. Such changes affect only the selected REST Web application. For most of these changes, you must disable the REST Web application before you make them, then enable the REST Web application afterward to evaluate the effect.

Manage OE Web Servers

You can manage OE Web Servers by:

- Start and stop Tomcat server on page 172
- Create an OE Web Server instance on page 173

- Configure an OE Web Server instance on page 174
- Edit an OE Web Server instance configuration on page 175
- Set OE Web Server application defaults on page 176
- Deploy a REST Web application (.war file) on page 176
- View the status of an OE Web Server instance on page 177
- View OE Web Server instance statistics on page 177
- Change OE Web Server run-time properties on page 178
- Delete an OE Web Server instance on page 179
- Enable or disable an OE Web Server instance on page 179
- View the OE Web Server log file on page 179

Each of these tasks is described in the following topics.

Start and stop Tomcat server

OpenEdge is shipped with a Tomcat server. The installed and configured Tomcat server provides a Java AdminServer and a web server for the REST Web application development and testing. The Tomcat server is available in your OpenEdge installation directory, $DLC/servers/tomcat.

Before you define/publish an OE Web Server or REST Service, you must start the Apache Tomcat server. You can start Apache Tomcat server installed with OpenEdge either using the Resources panel in the grid frame, or using Proenv.

1. To start the Tomcat server using the Resources panel:
   a) Click **Resources** in the management console menu bar. The resource types appear in the list frame.
   b) Expand the **Web Server** category, and click on the Tomcat instance. The **Details** page for that instance appears in the detail frame.
   c) In the **Command and control** section of the page, click **Control**. The Apache Tomcat installed with OpenEdge page opens.
   d) Click on the Start Web Server button to start Tomcat server.

   Note: The button is a Toggle button depending on the status of the Tomcat server instance. If the Web server is already started, the button displays the Stop option. To stop the Web server that is already running, click the Stop Web Server button.

2. To start the Tomcat server using Proenv:
   a) On the Windows Start > Programs menu, select Progress > OpenEdge > Proenv. Proenv opens a command prompt and configures it for your OpenEdge installation, displaying the configuration:

      ```
      proenv>
      ```

   b) Enter the following command:

      ```
      protc start
      ```

      A Tomcat shell window opens displaying the status of the server as running.

Note: To stop the Tomcat Server, enter **`protc stop`** at the **`proenv>`** command prompt.

Create an OE Web Server instance

You can create new OE Web Server instances from the **Resources** tab.

Note: Make sure that the REST Management Agent is deployed before you create the OE Web Server instance. You must copy the oerm.war file to the deployment server.

To create a new OE Web Server instance:

1. From the management console bar, click the **Resources** tab's dropdown arrow. Choose **New OpenEdge Resource** > **OE Web Server**. The **New OE Web Servername** page appears.
2. Type the name of the new OE Web Server in the **New OE Web Servername** field.

   Note: The OE Web Server name is case sensitive and can include any character except a period (.) or square brackets ([ ]). The name must be unique among all configured OE Web Server names.

3. Type the URL in the URLfield. You can now configure the properties of the new instance. Initially, the instance is enabled only for administration, allowing you to modify its properties and deploy REST services to it without allowing premature user access to an incomplete configuration. Form the root URL according to the following format:

   ```
   http://hostmachine:port
   ```

   Where:

   - **`hostmachine`** — The name of the machine where the OEM is installed (Default: localhost).
   - **`port`** — The HTTP port number on the machine (Default: 8089).

4. Under Deploy OE Web Server section, you can choose to deploy the OE Web Server instance on the Tomcat Manager (Apache Tomcat installed along with the Progress Developer Studio for OpenEdge).

   Note: Ensure that you have Tomcat Manager API enabled and a Tomcat user setup to enable this functionality.

   - **`Deploy OE Web Server`** — Select this check box to deploy the OE Web Server instance. Selecting this option requires you to enter the Tomcat Manager Details fields: Username, Password, and URL.
     - **`username`** — A username required to access the Tomcat Server.
     - **`password`** — The password for the specified username.
     - **`URL`** — The URL where you have the Tomcat Server installed. For example, if you are using the default ports in the Tomcat server, you can enter the URL as `http://localhost:8980/manager/text/deploy`.

Note: You can choose to create an OE Web Server instance without deploying the OE Web Server.

5. Click Save. The new OE Web Server instance appears under the OE Web Server folder.

Configure an OE Web Server instance

Before deploying REST services, you must configure an OE Web Server instance to host the services that you want to deploy or import. To configure an OE Web Server instance, you set properties for it. After the OE Web Server instance is configured, you can deploy a service and configure additional properties that affect execution of the service.

To configure an OE Web Server instance, you can view or edit properties in the following categories:

- Location (read)
- Proxy Server Setting (read and edit)
- Security (read and edit)

Set OE Web Server instance properties

You can set location and proxy server setting for an OE Web Server instance.

To set OE Web Server instance properties:

1. Click **Resources** in the management console menu bar. The resource types appear in the list frame.
2. Expand the **OE Web Server** category, and click on an instance. The **Details** page for that instance appears in the detail frame.
3. In the **Command and control** section of the page, click **Configuration**. The **OE Web Server Configuration** page opens. From this page, you can:
   - View and display of the OE Web Server properties.
   - Click **Edit** to modify the properties. For details, see Edit an OE Web Server instance configuration on page 175.

OE Web Server Location properties

You set the properties, shown in the following table, in this category when you first create the OE Web Server instance. For more information see Create an OE Web Server instance on page 173.

Table 82: OE Web Server Location properties

Property	Description
Location	Indicates that the machine on which the OE Web Server is installed is remote.
URL	Specifies the URL for the OE Web Server. For example, http://localhost:8080/oerm

OE Web Server Proxy Server Setting properties

The properties in this category, as shown in the following table, apply to both local and remote OE Web Server instances. They specify the information required to access any proxy server for the network on which the OE Web Server resides.

Table 83: OE Web Server Proxy Server Setting properties

Property	Description
Proxy host	The TCP host name for the proxy server Installed Default: None
Proxy port	The TCP port number of the proxy service Installed Default: 0
Proxy username	A valid username required to access the proxy server Installed Default: None
Proxy password	The password for the specified username Installed Default: None

OE Web Server Security properties

The properties in this category, as shown in the following table, apply to both local and remote OE Web Server instances. They specify the information required to authenticate the type of user who will login to OE Web Server.

Table 84: OE Web Server Security properties

Property	Description
No user authentication	Access the OE Web Server without any user authentication (no username and password).
Use default authentication	Access the OE Web Server using the default username and password based on the Web Server authorization setting. This option is selected by default.
Login to Web Server with username and password	Access the OE Web Server using the new username and password provided by you. Selecting this option requires you to enter the username and password details in the Login page.

Edit an OE Web Server instance configuration

You can edit the configuration properties of an OE Web Server instance from the management console.

To edit OE Web Server properties:

1. Click **Resources** in the management console menu bar. The resource types appear in the list frame.
2. Expand the **OE Web Server** category, and click on an instance. The **Details** page for that instance appears in the detail frame.
3. In the **Command and control** section of the page, click **Configuration**. The **OE Web Server Configuration** page opens.
4. Click **Edit**, and make the changes you want to the properties.
5. Click **Save**.

Set OE Web Server application defaults

You can view and set the default values used to initialize properties for REST Web Applications that you deploy. Any changes to these defaults affect only REST Web Applications that you deploy after confirming the change. To set the values of properties for deployed REST Web Applications, set the property values individually for each REST Web application.

To view and set OE Web Server application defaults:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the OE Web Server instance whose default values you want to view.
3. Click the OE Web Server instance. The **OE Web Server** details page appears.
4. In the **Command and control** section of the page, click **Defaults**. The **OE Web Servers Defaults** page appears.
5. Click **Edit**.
6. Select the Runtime property whose default value you want to change. For details on the available properties, see *OpenEdge Application Server: Administration*.
7. Type the new value in the available field. For logical values, typing any value other than `true` results in a setting of `false`.
8. Click **Submit** to change the default value.

To return to the default values, you can click **Reset** from the Defaults page.

Deploy a REST Web application (.war file)

You can deploy a REST Web application (.war file) to the Servlet AdminServer where the OpenEdge Manager Servlet instance is installed.

A deployed REST Web application receives its initial property values from the `default.props` file for the REST instance. You can change these default property values after you deploy the .war file. A REST Web application always deploys with a disabled status to prevent premature or unintended client access.

To deploy a REST Web application:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the OE Web Server instance where you want to deploy the REST Web application.
3. Click the OE Web Server instance. The **REST Applications** page appears.
4. In the **REST Applications** section of the page, click the Deploy button.

5. In the Deploy REST Application .war dialog that comes up, browse for the path and name of the WAR (.war) file for the REST Web application that you want to deploy.
6. You can optionally provide the REST Web application name in the Application name field.
7. Click **Deploy** to deploy the REST Web application with the default settings. A confirmation message appears, and the WAR file name is identified in the **REST Applications** section of the REST Applications page.

Deployment generates files in the OE Web Server instance directory, one of which is `applicationName.props`, an XML file containing the current REST Web application property settings (initially set from `default.props`)

Caution: You must disable the REST Web application before you try to modify any of the properties. Two of the properties, serviceLogEntryLevel and serviceLoggingLevel can be modified even when the application is Enabled.

After deployment, to make the same REST Web application available using different information, deploy a new REST Web application with the new information, using a different REST Web application name in the same OE Web Server context. You could also deploy the REST Web application to a different OE Web Server context.

However, you can change the run-time properties of a deployed REST Web application at any time.

View the status of an OE Web Server instance

You can view the status of a selected OE Web Server instance to see if the instance is running. If the instance is running, the status indicates that it is enabled.

To view the status of an OE Web Server instance:

1. Click **Resources** in the management console menu bar. The resource types appear in the list frame.
2. Expand the **OE Web Server** category, and click on an instance. The **Details** page for that instance appears in the detail frame.
3. Under **Operations views**, click **Status**. The following status information appears:
   - Whether the OE Web Server instance is running
   - Whether access to administrative functions is enabled

   Note: You can also view the status of the OE Web Server instance on the top right corner of the **Details** page.

View OE Web Server instance statistics

You can view and reset run-time statistics for a selected OE Web Server instance. Each OE Web Server instance creates and begins accumulating these statistics, Start Time, when it is first started by the Java Servlet Engine and Creation Time. You can also reset the statistics at any point during OE Web Server execution, so that the statistics begin accumulating again from a new Start Time.

To view the statistics of an OE Web Server instance:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the OE Web Server instance whose statistics you want to view.

3. Click the OE Web Server instance. The **OE Web Server** details page appears.
4. Under **Operational Views**, click **Statistics**. The following information appears in the Statistics page:
   - **Active Requests** — Requests currently queued and being acted upon by the OE Web Server
   - **Authentication Errors** — Total number of authentication errors
   - **HTTP Request Errors** — Requests received from the HTTP listener, including administrative, and OE Web Server requests
   - **HTTP Requests** — Total requests received from the HTTP listener, including administrative, and REST requests
   - **Method not allowed errors** — Total errors returned by the OE Web Server, with error counts broken out at the bottom of the list for each of several error categories when total errors are greater than zero (0)
   - **REST Disabled errors** — calls while OE Web Server is disabled
   - **Services disabled** — REST Web Applications deployed to this OE Web Server that are disabled from client access
   - **URL not found errors** — calls made to unidentified resources
   - **Creation Time** — Date and time that counting began for the statistics
   - **Start Time** — Start Date and time of the OE Web Server instance
5. To reset the statistics and start accumulating all statistics from zero, click **Reset** f you want to reset the statistics.

Change OE Web Server run-time properties

You can temporarily change some OE Web Server properties at run time without restarting your Java Servlet Engine (JSE). This is most useful for testing and debugging. The next time you restart your JSE, these settings revert to the current configuration settings for these properties stored in the properties file for the OE Web Server. The OE Web Server property file is stored in the WEB-INF directory of the deployed OE Web Server.

To change OE Web Server run-time properties:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the OE Web Server instance whose run-time properties you want to change.
3. Click the OE Web Server instance. The **OE Web Server** details page appears.
4. Under **Operations views**, click **Run-time Properties**.
5. Click **Edit** to modify any of the following properties:
   - **adminEnabled** — Specifies whether administrator privileges are in effect, either **enabled o**r **disabled**.
   - **description** — Specifies the description of the OE Web Server instance.
   - **logAppend** — Specifies whether to start a new log file each time the server is restarted, either true or false.
   - **logEntryTypes** — Specifies a list of logging entry types separated by commas.
   - **logFile** — Specifies the location and name of the OE Web Server log file.
   - **loggingLevel** — Specifies the amount of information to be written to the broker log.
   - **managerPropertiesURI** — Specifies the identifier for the OE Web Server resource.
   - **name** — Specifies the name of the OE Web Server.
   - **propertiesFileName**— Specifies the path and name of the oerm.properties file.

- **webAppEnabled** — Controls the OE Web Server's ability to accept and process requests to any of its deployed OE Web Server applications.
- **workDir** — Specifies the path of the OpenEdge working directory.

6. Click **Submit**.

Delete an OE Web Server instance

You can delete any OE Web Server instance listed in the management console as long as it has no REST services deployed to it.

To delete an OE Web Server instance:

1. Click **Resources** in the management console menu bar. The resource types appear in the list frame.
2. Filter, or search for, the OE Web Server instance you want to delete.
3. Click the OE Web Server instance. The **OE Web Server**details page appears.
4. Click **Delete**, and click **OK** to confirm the deletion.

   The **OpenEdge Management Resources** page appears. A set of next-step options that are related to resources are available for your use.

Note: You can choose to stop the OE Web Server before deleting it. However, all the running applications get undeployed when you delete an OE Web Server.

Note: Deleting the OE Web Server instance does not undeploy it from the application server. To undeploy the OE Web Server instance from the application server, you must uninstall the OE Web Server instance from the application server after removing the instance from OEM.

Enable or disable an OE Web Server instance

From the management console, you can enable or disable monitoring of an OE Web Server instance.

To enable or disable monitoring of an OE Web Server instance:

1. Click **Resources** in the management console menu bar. The resource types appear in the list frame.
2. Expand the **OE Web Server** category, and click on an instance. The **Details** page for that instance appears in the detail frame.
3. Click **Control**. That instance's OE Web Server page appears.
4. Click **Edit**.
5. Select the **Enabled** check box to enable the OE Web Server monitoring or clear the check box to disable it.

View the OE Web Server log file

You can view the log file for an OE Web Server by using the log file viewer. The log file viewer allows you to examine the log file through an HTML interface.

To access the log file viewer:

1. Click **Resources** in the management console menu bar. The resource types appear in the list frame.
2. Expand the **OE Web Server** category, and click on an instance. The **Details** page for that instance appears in the detail frame.
3. Click **Log File Viewer**. The **OE Web Server Log File Contents** page opens. You can use the Log File Viewer in the following ways:

   - Use the **Show** field to control how many database log file entries display at one time. The number entered into the **Show** field cannot be less than 10.
   - Use the **Overlap** field to control how many entries are repeated from screen to screen.

   Note: The value in the **Overlap** field cannot be more than the number in the **Show** field minus one. For example, if you show 30 entries, you can overlap only 29 or fewer of them.

   - Choose **Descending** to change the default display that is in ascending order. Note that the **Show** field dictates the number of entries shown, regardless of whether they display in ascending or descending order.
   - Click **Reload** after you have changed any values for the number of lines or lines per page. If you do not reload, the viewer continues to display the previous values.
   - Click **Go To** to control the numbered entry in the log file from where the viewer begins its display. For example, a value of 10 entered into the **Go To** field will begin the display from the tenth log file entry.
   - Click **First** to display the first x entries, where x is the value in the **Show** field.
   - Click **Prior** to display the previous x entries, where x is the value in the **Show** field.
   - Click **Next** to display the next x entries, where x is the value in the **Show** field.
   - Click **Last** to display the last x entries, where x is the value in the **Show** field. If the contents of the log file have changed since you opened the viewer, the log file viewer indicates this in the **Log file status** field.

Work with REST Web Applications

You can perform the following REST Web application administration tasks from the management console:

- Enable or disable a REST Web application for client access on page 180
- View REST Web application statistics on page 181
- Use the REST Web application log file viewer on page 182
- Update a deployed REST Web application on page 183
- TLS support for REST Web Applications on page 184
- Undeploy a REST Web application on page 184

Enable or disable a REST Web application for client access

You can enable any deployed REST Web application, making it accessible or inaccessible to user requests on the network. A deployed REST Web application is disabled by default.

To enable or disable a REST Web application for client access:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the OE Web Server instance where the REST Web application you want to enable or disable is deployed.
3. Click the OE Web Server instance. The **OE Web Server** Details page appears.
4. Click the REST Applications link. The REST Applications page appears.
5. Under the **REST Applications** section, select the deployed REST Web application.
6. Click **Enable in the Actions column to enable the application. The Enabled column** displays the status of the **Enabled** option.
7. To disable the enabled application, click **Disable** in the **Actions** column.

View REST Web application statistics

You can view (and reset) run-time statistics for a selected REST Web application. Each REST Web application creates and begins accumulating these statistics (Start Time) from the time that its parent OE Web Server instance starts up (Creation Time). You can reset the REST Web application statistics at any point while the OE Web Server instance is running. A reset clears and begins accumulating the statistics again with a new Start Time.

To view the run-time statistics for a REST Web application:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the OE Web Server instance where the REST Web application whose runtime statistics you want to view is deployed.
3. Click the OE Web Server instance. The **OE Web Server** details page appears.
4. Click the REST Applications link. The REST Applications page appears.
5. Under the **REST Applications** section, select the deployed REST Web application.
6. Under **REST Application Details section**, click the Statistics tab to view a read-only statistics presentation with the following information:
   - **Adapter errors** — Total number of OE Web Server failures
   - **Application errors** — Total number of errors related to REST Web application
   - **Average response time** — Average response time of each request response.
   - **Failed admin logins** — Total number of failed administrator logins
   - **Failed admin requests** — Total number of administrator requests that failed due to missing request information
   - **Failed app server connections** — Total number of failed AppServer connections in the connection pool
   - **Failed user logins** — Total number of failed user logins
   - **Failed user requests** — Total number of requests that failed due to missing request information
   - **Max response time** — Maximum response time of each request response
   - **Successful admin logins** — Total number of successful administrator logins
   - **Successful admin requests** — Total number of successful administrator requests

- **Successful app server connections** — Total number of successful AppServer connections in the connection pool for this REST Web application
- **Successful user logins** — Total number of successful user logins
- **Successful user requests** — Total number of successful user requests handled by the selected REST Web application
- **Creation time** — Date and time that counting began for the statistics
- **Start time** — Start date and time of the REST Web application

7. To reset the statistics and start accumulating from zero, click **Reset** if you want to reset the statistics.

Use the REST Web application log file viewer

To view the contents of a REST Web application log file, access the viewer associated with each individual log file.

The log file viewer allows you to examine the contents of a REST Web application log file in HTML format. You can access a log file viewer from the following location:

- Click the **Log File Viewer** button in the **REST Applications** section of the **REST Applications** page. The log file viewer button appears with every deployed application.

The following information will help you use the REST Web application log file viewer:

- Use the **Show** field to control how many OE Web Server log file entries appear at one time. The number entered into the **Show** field cannot be less than 10.
- Use the **Overlap** field to control how many entries are repeated from screen to screen.

  Note: The value in the **Overlap** field must not be more than the number in the **Show** field minus one. For example, if you show 30 entries, you can overlap only 29 or fewer of them.

- Click **Reload** after changing the values in either the **Show** field or **Overlap** field. OpenEdge Management prompts you to click **Reload**. A warning message that reads **changed, reload needed** appears in the **File log status** field in the **log file summary** section of the page.

  If you do not reload, the viewer displays the previous values.

- Click **Go To** to control which numbered entry in the log file the viewer begins its display with. For example, a value of `10` entered into the **Go To** field begins the display from the tenth log file entry.

  Note: You must click **Go To** after entering a value in the **Go To** field, or the viewer will not update its display.

- The default display of entries is in ascending order. Choose **Descending** to change the display. Note that the **Show** field dictates the number of entries shown, whether they appear in ascending or descending order.
- Click **First** to display the first *x* entries, where *x* is the value in the **Show** field.
- Click **Prior** to display the previous *x* entries, where *x* is the value in the **Show** field.
- Click **Next** to display the next *x* entries, where *x* is the value in the **Show** field.
- Click **Last** to display the last *x* entries, where *x* is the value in the **Show** field.
- To view additional log file entries without changing your current starting log file entry, leave the **Go To** field blank, change the value in the **Show** field, and click **Reload**.

Update a deployed REST Web application

You can update the definition and deployment information for a deployed OE Web Server at any time.

Caution: You can freely update the definition and deployment information for a deployed REST Web application during development. However, once you deploy and enable a REST Web application for client access, avoid making any changes to this information, as client implementations depend on its stability.

After deployment, to make the same REST Web application available using different information, deploy a new REST Web application with the new information, using a different REST Web application name in the same OE Web Server context. You can also deploy the REST Web application to a different OE Web Server context.

Set OE Web Server application properties

You can view and set the value of each property for a deployed REST Web application, or you can reset the values of all properties to the current REST Web application defaults (stored in the `default.props` file for the OE Web Server instance where the REST Web application is deployed).

You must disable a REST Web application before you can set most properties for it. You can verify if the REST Web application is disabled by checking its status. The only properties that you can set for an enabled OE Web Server are:

- **serviceLogEntryLevel**
- **serviceLoggingLevel**

If you set other properties while the REST Web application is enabled, the property value changes take effect only after you disable and then enable the REST Web application again.

Note: Unlike OE Web Server instance properties, the values for REST Web application properties are not stored in the `ubroker.properties` file. They are stored in the `applicationname.props` XML file located in the Web application Servlet directory for the OE Web Server instance where the REST Web application is deployed.

To view and set REST Web application properties:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the OE Web Server instance where the REST Web application whose properties you want to view and set is deployed.
3. Click the OE Web Server instance. The **OE Web Server** details page appears.
4. Click the REST Applications link. The REST Applications page appears.
5. Under the **REST Applications** section, select the deployed REST Web application.
6. Under **REST Application Details section**, select the Properties tab to view the application properties of the deployed REST Web application.

   For details on the properties that can appear in this list, see *OpenEdge Application Server: Administration*.

   Note: The actual list of properties displayed is a subset of all the properties, and changes depending on whether the session model of the OE Web Server is session managed or session free. To verify the session model of an OE Web Server, check its status. For more information, see Enable or disable a REST Web application for client access on page 180.

7. Type the new value in the available field. For logical values, entering any value other than true results in a setting of false.
8. Click **Save** when you finish. The changes are then written to the `applicationname.props` file for the selected REST Web application.

TLS support for REST Web Applications

To have a secure connection between the REST Web application and the AppServer, the following conditions must be met:

- You must obtain and install public key certificates for the host machine in which the OE Web Server resides.

  Note: You can add any additional public certificates to psccerts.jar (located in WEB-INF/lib folder and contains all the certificates for the REST Web application required to connect to the OpenEdge AppServer) using the certutil.bat. If you do not want to use pscerts.jar, you can uncomment the property certlocation in the REST Web application's `web.xml file`.

- The REST Web application must send TLS requests to the AppServer to process the client requests. To configure the service to send TLS requests, set the value of the **ApplicationService Protocol** property to AppServerS or AppServerDCS. For more information on setting properties for REST Web Applications, see Set OE Web Server application properties on page 183 .
- The AppServer must be TLS-enabled, meaning that it accepts TLS requests from the OE Web Server (or other clients). OpenEdge supports Transport Layer Security (TLS)-enabled communications between the Web server where the REST Management Agent runs and a Web server client. To achieve this, you must enable the REST Web application for HTTPS (TLS) connections. For more information on enabling TLS, see Viewing or modifying AppServer broker properties.

For details on TLS support in OpenEdge, including configuring and operating a client of an TLS-enabled AppServer, see *OpenEdge Getting Started: Core Business Services - Security and Auditing*.

Undeploy a REST Web application

You can undeploy a deployed REST Web application, which removes its deployment information, its properties, and other settings from the OE Web Server instance context.

You must disable a REST Web application before you can undeploy it. You can verify if the REST Web application is disabled by checking its status.

To undeploy a REST Web application:

1. Click **Resources** in the management console menu. All resources managed by your console appear in the grid frame.
2. Filter or search for the OE Web Server instance where the REST Web application you want to enable or disable is deployed.
3. Click the OE Web Server instance. The **OE Web Server** details page appears.
4. Click the REST Applications link. The REST Applications details page appears.
5. Under the **REST Applications** section, select the deployed REST Web application.
6. Click **Undeploy in the OE Web Server's column to undeploy the application.**

The application is undeployed. If the REST Web application is already disabled, it is now undeployed from the OE Web Server instance context and removed from the management console. If the REST Web application is still enabled, an error appears indicating that you must disable the REST Web application before undeploying it.

Note: You can undeploy multiple REST Web Applications at the same time, by selecting the checkbox against the application name and clicking the Undeploy button on the top left corner of the **REST Application section**.

9

Configure Batch Programs

The following topics describe how to configure and manage Batch Program resources using OpenEdge Management.

For details, see the following topics:

- Overview
- Create a Batch Program
- Modify Batch Program configuration
- Start a Batch Program
- Stop a Batch Program
- Delete a Batch Program
- View and manage Batch Program processes
- View Batch Program log files
- Batch Program properties
- API references

Overview

A Batch Program is an OpenEdge resource designed to run for longer periods of time and supports separate scripts to start, monitor, and stop the Program. You can manage and monitor Batch Program resources using OpenEdge Management.

Each Batch Program resource can have several copies of the program that is running. OpenEdge Management considers each program as an individual resource and allows you to start, stop, and monitor it. OpenEdge Management monitors the Batch Program instances by defining rules for each instance and generating alerts based on the status of the program.

Like any other resource, OpenEdge Management allows you to create, configure, and monitor Batch Program resources. When configuring a Batch Program resource, you can write scripts to monitor the processes supported by OpenEdge Management. Also, you can write your own scripts to monitor the processes that are NOT supported by OpenEdge Management. Once configured, they appear in the OpenEdge Management Resource Grid page.

Note: Creating, managing, and monitoring of Batch Program resources is not supported in OpenEdge Explorer.

Create a Batch Program

You can create a Batch Program resource using the **Create Batch Program** page in the OpenEdge Management.

To create a Batch Program:

1. From the OpenEdge Management console menu, click **Resources** > **New** > **Batch Program**.

   The **Create Batch Program** page appears.
2. Specify the configuration properties as described in Batch Program properties on page 192.
3. Click **CREATE**.

   The **Batch Program: <instance name>** page appears with **Command and Control** and **Processes** sections.

Modify Batch Program configuration

You can modify the configuration of a Batch Program resource using the **Edit Batch Program** in the OpenEdge Management.

To modify the configuration of a Batch Program resource:

1. From the OpenEdge Management console menu, click **Resources** > **Go to Resources**.

   All the resources managed by your console appear in a grid frame.
2. Filter or search for, and select the required Batch Program resource.

   The **Batch Program: <instance name>** page appears.

3. In the **Command and Control** section, click **Configuration**.

   The **Edit Batch Program** page appears.

4. Click **EDIT** and then modify the Batch Program resource properties as described in Batch Program properties on page 192.

5. Click **SAVE**

Start a Batch Program

You can start a Batch Program resource using the **Batch Program: <instance name>** in the OpenEdge Management.

To start a Batch Program resource:

1. From the OpenEdge Management console men, click **Resources** > **Go to Resources**.

   All the resources managed by your console appear in a grid frame.

2. Filter or search for, and select the required Batch Program resource.

   The **Batch Program: <instance name>** page appears.

3. Click **START PROGRAM**.

   The Batch Program process is started and appears in the **Processes** section.

   The **START PROGRAM** option is displayed only when:

   - The Startup script is specified as described in Batch Program Startup properties on page 197.
   - The Batch Program is not running and supports only one process.
   - The Batch Program is running already but supports multiple processes.

4. To run multiple processes, click **START PROGRAM** again.

Stop a Batch Program

You can stop a Batch Program resource using the **Batch Program: <instance name>** in the OpenEdge Management.

To stop a Batch Program resource:

1. From the OpenEdge Management console men, click **Resources** > **Go to Resources**.

   All the resources managed by your console appear in a grid frame.

2. Filter or search for, and select the required Batch Program resource.

   The **Batch Program: <instance name>** page appears.

3. Click **STOP BATCH PROGRAMS**.

4. Click **YES** in the confirmation dialog box to stop all the copies of the Batch Program resource..

   The **Batch Program Control** page appears.

This page displays the number of processes running in the **Program summary** section. When the processes are stopped, the Process Count is displayed as 0.

Optionally, you can click **START BATCH PROGRAM** to start the Batch Program resource again.

5. To stop an individual process, see View and manage Batch Program processes on page 190.

Delete a Batch Program

You can delete a Batch Program resource using the **Batch Program: <instance name>** in the OpenEdge Management.

To delete a Batch Program resource:

1. From the OpenEdge Management console men, click **Resources** > **Go to Resources**.

   All the resources managed by your console appear in a grid frame.

2. Filter or search for, and select the required Batch Program resource.

   The **Batch Program: <instance name>** page appears.

3. Click **DELETE**.

   Note: Ensure that you stop all the Batch Program processes before deleting the Batch Program. Clicking **DELETE** does not stop the Batch Program processes that are running.

4. Click **YES** in the confirmation dialog box.

View and manage Batch Program processes

You can view and manage multiple processes that are starting, running, or stopped for a Batch Program resource using the **Batch Program: <instance name>** page in the OpenEdge Management. This enables you to monitor the status and other properties of multiple Batch Program processes simultaneously, and if required, stop or delete a process.

To view the available processes for a Batch Program resource:

1. From the OpenEdge Management console men, click **Resources** > **Go to Resources**.

   All the resources managed by your console appear in a grid frame.

2. Filter or search for, and select the required Batch Program resource.

   The **Batch Program: <instance name>** page appears.

3. In the **Processes** section, you can view the following information for each Process:

Property	Description
PID	The Process ID (PID) of the process.
Name	The name of the process

Property	Description
State	The status of the process – **Starting**, **Running**, or **Not Running**.
Message	Provides additional information such as Process timed out or PID not alive.
Start Time	The time when the process is started.
Stop Time	The time when the process is stopped.
Up Time	The time indicating how long the process has been running as reported by the operating system. This is a text entry and can be weeks, days, minutes, or seconds.
CPU	The percentage of CPU utilized by the process.
Memory	The amount of memory consumed by the process.
Log Files	The links to view the standard out and err files in the Log File Viewer.

4. To stop an individual process that is running, click the **Stop** icon adjacent to the PID of that process.

   If a process is not stopped using the **Stop** icon, click the PID link of that process and then click **KILL** to terminate it. Click **YES** in confirmation dialog box and then click **OK** to return to the **Batch Program: <instance name>** page.

   Note: Clicking **STOP BATCH PROGRAMS** in the **Batch Program: <instance name>** page stops all the processes that are running.

5. To delete a process that is not running, click the **Delete** icon adjacent to the PID of that process.

View Batch Program log files

You can view the log files—stdout and stderr—of a Batch Program resource using the log file viewer in the OpenEdge Management. This allows you to examine the contents of the Batch Program resource log files through an HTML interface.

To view the Batch Program resource log files:

1. From the OpenEdge Management console men, click **Resources** > **Go to Resources**.

   All the resources managed by your console appear in a grid frame.

2. Filter or search for, and select the required Batch Program resource.

   The **Batch Program: <instance name>** page appears.

3. In the **Command and Control** section, you can choose to click one of the following to view the log file viewer:

   - **Log File Viewer (stdout)** — Displays the standard out log file entries.

- **Log File Viewer (stderr)** — Displays the standard err log file entries.

4. In the **Log File Viewer**, you can:
   - Click **FIRST** to display the first **x** entries, where **x** is the value in the **Show** field.
   - Click **PRIOR** to display the previous **x** entries, where **x** is the value in the **Show** field.
   - Click **NEXT** to display the next **x** entries, where **x** is the value in the **Show** field.
   - Click **LAST** to display the last **x** entries, where **x** is the value in the **Show** field.
   - Click **RELOAD** to reload the values, if changed in either the **Show** field or the **Overlap** field.
   - Use the **Go to** field to type the number of the log file entry you want the log file viewer to begin with.

     For example, if you type a value of 10 in the **Go to** field and press **Enter**, the log file viewer displays from the tenth log file entry.
   - Use the **Show** field to type the number of log file entries you want the log file viewer to display at one time.

     The value in the **Show** field cannot be less than 10.
   - Use the **Overlap** field to type the number of log file entries you want to be repeated from screen to screen.

     The value in the **Overlap** field cannot be more than the number in the **Show** field minus one. For example, if you show 30 entries, you can overlap only 29 or lesser.
   - Use the **Display** drop-down to choose the order in which you want the log file entries to be displayed, **Ascending** or **Descending**. The default is **Ascending**.
   - If the contents of the log file have changed since you opened the viewer, the log file viewer indicates this in its footer's **Status** field.

Batch Program properties

When creating or configuring a Batch Program resource, you can specify or modify the Batch Program properties as described in the following sections:

- Batch Program General properties on page 196
- Batch Program Startup properties on page 197
- Batch Program Monitoring properties on page 200
- Batch Program Shutdown properties on page 203
- Batch Program Environment variable properties on page 204

Table 85: Startup properties

Property	Description
Startup script	(Optional) The name of the Batch Program that you want to run. This can be the Batch Program or a script that starts the Batch Program.
Startup script parameters	(Optional) The parameters that are to be passed to the Startup script.
Startup script timeout (seconds)	(Optional) The timeout value for the Batch Program to start. The default value is 15 seconds. If the status of the Batch Program is unknown within the timeout value, OpenEdge Management considers it as a failure.
Kill if still running after timeout	Terminates the Startup script, if running, after the specified timeout. If this option is not selected and the Startup script is not terminated after the specified timeout, OpenEdge Management tracks the PID of the script.
Supports multiple processes	(Optional) Supports running multiple process.
Startup failure behavior • On first startup failure • On second startup failure • On subsequent startup failure(s)	(Optional) The behavior of OpenEdge Management when the startup of the Batch Program fails on the first, second, and subsequent startups. Select one or more behaviors from the following options: • **Fire alert** — Fires an alert. • **Perform action** — Executes an action. • **Restart the program** — Restarts the Batch Program. The default is that the OpenEdge Management does not follow any behavior.
Subsequent startup limit	(Optional) The number of times the Subsequent startup behavior must be executed. The minimum value is 1 and there is no maximum value. The program restarts until it reaches the specified limit.

Table 86: Monitoring properties

Property	Description
Monitoring method	The method to track the status of the program. Select from the following options: • **Status script** — Tracks the status of the Batch Program. The status can be running or failed. The script can call other programs such as reading log files, reading PID files, or making an HTTP request. • **PID file** — The Process ID (PID) of the Batch Program. When you define the path for the PID file, OpenEdge Management looks for the PID file and considers that the Batch Program is running only if the file is valid and the process associated with PID is executing. • **PID registration** • **Exit code**
Status script	(Optional) The path to the Status script. OpenEdge Management utilizes this script to determine the status of the Batch Program.
Status script parameters	(Optional) The parameters that are to be passed to the Status script.
Status script timeout (seconds)	(Optional) The timeout value for the OpenEdge Management to wait for tracking the Batch Program's status. The default value is 15 seconds. The Status script must execute and terminate within the timeout value. If the Status script terminates with an exit code 0, OpenEdge Management considers that the status of the Batch Program as running. If the script terminates with an exit code other than 0, OpenEdge Management considers that the status as failed. If the script is missing or cannot be executed, OpenEdge Management considers that the Program is not running and failed.
Kill if still running after timeout	Terminates the Status script, if running, after the specified timeout. If this option is not selected and the Status script is not terminated after the specified timeout, OpenEdge Management tracks the PID of the script.
PID file	The absolute path to the location where PID is maintained by Startup or Shutdown scripts.
Status failure behavior	The behavior of OpenEdge Management when the status of the Batch Program is unknown. Select one or more behaviors from the following options: • **Fire alert** — Fires an alert. • **Perform action** — Executes an action. • **Restart the program** — Restarts the Batch Program.

Table 87: Shutdown properties

Property	Description
Shutdown method	The method to stop the Batch Program. Select from the following options: • **Shutdown script** • **Terminate** If you select **Terminate**, OpenEdge Management looks for the Process ID (PID) of the Batch Program and stops the Batch Program. If the PID is not available, OpenEdge Management fails to stop the Batch Program.
Shutdown script	(Optional) The path to the script to stop the Batch Program.
Shutdown script parameters	(Optional) The parameters that are to be passed to the Shutdown script.
Shutdown script timeout (seconds)	(Optional) The timeout value for the OpenEdge Management to wait for executing the Shutdown script to stop the Batch Program. The default value is 15 seconds. If the Shutdown script is not executed within the timeout value, OpenEdge Management considers it as a failure.
Kill if still running after timeout	Terminates the Shutdown script, if running, after the specified timeout. If this option is not selected and the Shutdown script is not terminated after the specified timeout, OpenEdge Management tracks the PID of the script.

Table 88: Environment variable properties

Property	Description
User	(Optional) The name or ID of the user account on the machine where the process is to be executed. This field allows you to determine the permissions that the process must have when executed.
Group	(Optional) (UNIX only) The name of the group.
Password	(Optional) The user account's password for authentication. It is stored in the configuration database.
Environment variables	(Optional) The environment variables specified by the user. These variables are provided during the execution of the Startup, Status, and Shutdown scripts.
Debug program	Debugs the Batch Program.

Batch Program General properties

Table 89: General properties

Property	Description
Name	A unique name for the Batch Program. **Note:** The name must not contain spaces or special characters such as an asterisk (*), an ampersand (&), or a period.
AdminServer	The AdminServer on which the Batch Program is to be executed. The default is the `localhost`. **Note:** Running a Batch Program resource on a remote AdminServer is not supported in OpenEdge 11.7.
Description	(Optional) The description for the Batch Program.
Enabled	(Optional) Enables the Batch Program for monitoring. This check box is not selected by default for testing purposes. Selecting this check box enables OpenEdge Management to perform status checks, trigger alerts, and execute actions based on the status of the Batch Program.
Working directory	(Optional) The working directory of the Batch Program. Specify a valid pathname. If not specified, the default is the OpenEdge Work directory.
Output file (stdout)	(Optional) The path where the standard out file from the Startup, Status, and Shutdown scripts is directed to. If not specified, the output from the Program will be written to the AdminServer log file.
Output file (stderr)	(Optional) The path where the standard err file from the Startup, Status, and Shutdown scripts is directed to. If not specified, the output from the Program will be written to the AdminServer log file.

Batch Program Startup properties

Table 90: Startup properties

Property	Description
Startup script	(Optional) The name of the Batch Program that you want to run. This can be the Batch Program or a script that starts the Batch Program.
Startup script parameters	(Optional) The parameters that are to be passed to the Startup script.
Startup script timeout (seconds)	(Optional) The timeout value for the Batch Program to start. The default value is 15 seconds. If the Startup script terminates within the specified timeout with a non-zero Exit code, OpenEdge Management considers it as a failure. If the Startup script is not terminated within the specified timeout, OpenEdge Management considers it as running.
Kill if still running after timeout	Terminates the Startup script, if running, after the specified timeout. If this option is not selected and the Startup script is not terminated after the specified timeout, OpenEdge Management tracks the PID of the script.
Supports multiple processes	(Optional) Supports running multiple process. If this option is not selected, you cannot start the multiple processes of the Batch Program neither from the OpenEdge Management console nor the REST API.
Startup failure behavior • On first startup failure • On second startup failure • On subsequent startup failure(s)	(Optional) The behavior of OpenEdge Management when the startup of the Batch Program fails on the first, second, and subsequent startups. Select one or more behaviors from the following options: • **Fire alert** — Fires an alert. • **Perform action** — Executes an action. • **Restart the program** — Restarts the Batch Program. To enable the behavior options on the next failure, select the **Restart the Program** option. The default is that the OpenEdge Management does not follow any behavior.
Subsequent startup limit	(Optional)) The maximum number of times the Batch Program can be started before the OpenEdge Management considers the Startup as a failure. The program restarts until it reaches the specified limit. The minimum value is 1 and the maximum value is 300. In a case where the Startup program continuously fails, and **Restart the Program** is selected to avoid busy loops, there will be a delay of few seconds between each attempt to restart.

Property	Description
Exit code	Indicates the status of the process when it exited. If the Startup script does not terminate within the specified timeout value, OpenEdge Management monitors the script until it exits and uses the exit code of the process to determine the action to perform. **Note:** For OpenEdge Management to monitor the exit code, the process should be started and should NOT be restarted before the process terminates. In a case where OpenEdge Management is restarted or the exit code is unknown, OpenEdge Management performs the default exit code action.
Action	The action that you want to associate with the failure of the Batch Program. For more information, see Batch Program default actions on page 204. Once you specify the exit code and select its associated action, click **ADD/UPDATE** to add them. You can choose to add multiple exit codes and their associated actions. Click **DELETE** to delete your selection.

Batch Program Monitoring properties

Table 91: Monitoring properties

Property	Description
Monitoring method	The method to track the status of the program. Select from the following options: • **Status script** — Tracks the status of the Batch Program. The script can perform any action to determine the Batch Program's return state including reading log files, reading PID files, or making an HTTP request. This script can be Windows batch programs or executables, or UNIX shell scripts or executables. • **PID file** — The file that contains a Process ID (PID) of the Batch Program. If available, OpenEdge Management uses this file to determine the state of the process. When you define the path for the PID file, OpenEdge Management looks for the PID file and considers that the Batch Program is running only if the file is valid and the process associated with PID is executing. In a case where the PID file is not available, cannot be read, or does not contain a single line with an integer value, the status is considered as failed. • **PID registration** — Exposes a REST API with which a stand-alone process can register a PID with OpenEdge Management. OpenEdge Management monitors the PID periodically to ensure that it is running. For the process to be monitored, it must be running on the same machine where the AdminServer is configured for the Batch Program. • **Exit code** — Indicates the status of the process when it exited.
Status script	(Optional) The path to the Status script. OpenEdge Management utilizes this script to determine the status of the Batch Program.
Status script parameters	(Optional) The parameters that are to be passed to the Status script.
Status script timeout (seconds)	(Optional) The timeout value for the OpenEdge Management to wait for tracking the Batch Program's status. The default value is 15 seconds. The Status script must execute and terminate within the timeout value. If the Status script terminates with an exit code 0, OpenEdge Management considers that the status of the Batch Program as passed. If the script terminates with an exit code other than 0, OpenEdge Management considers that the status as failed. If the script is missing or cannot be executed, OpenEdge Management considers that the Program is not running and failed.
Kill if still running after timeout	Terminates the Status script, if running, after the specified timeout. If this option is not selected and the Status script is not terminated after the specified timeout, OpenEdge Management writes a message to the Batch Program log files.

Property	Description
PID file	The absolute path to the location where PID is maintained by Startup or Shutdown scripts. However, the path can also be a relative path or to OpenEdge work directory (`%DLC%\WRK`).
Status failure behavior	The behavior of OpenEdge Management when the status of the Batch Program is unknown. Select one or more behaviors from the following options: • **Fire alert** — Fires an alert. • **Perform action** — Executes an action. • **Restart the program** — Restarts the Batch Program.
Action	The action that you want to associate with the failure of the Batch Program. For more information, see Batch Program default actions on page 204. Once you specify the exit code and select its associated action, click **ADD/UPDATE** to add them. You can choose to add multiple exit codes and their associated actions. Click **DELETE** to delete your selection.

Batch Program Shutdown properties

Table 92: Shutdown properties

Property	Description
Shutdown method	The method to stop the Batch Program. Select from the following options: • **Shutdown script** • **Terminate** If you select **Terminate**, OpenEdge Management looks for the Process ID (PID) of the Batch Program and stops the Batch Program. If the PID is not available, OpenEdge Management fails to stop the Batch Program. The **Terminate** option works only if PID file or PID registration is used for status monitoring, or the startup program is running even after the specified timeout value and exit code is selected for status monitoring.
Shutdown script	(Optional) The path to the script to stop the Batch Program.
Shutdown script parameters	(Optional) The parameters that are to be passed to the Shutdown script.
Shutdown script timeout (seconds)	(Optional) The timeout value for the OpenEdge Management to wait for executing the Shutdown script to stop the Batch Program. The default value is 15 seconds. If the Shutdown script is not executed within the timeout value, OpenEdge Management considers it as a failure.
Kill if still running after timeout	Terminates the Shutdown script, if running, after the specified timeout. If this option is not selected and the Shutdown script is not terminated after the specified timeout, OpenEdge Management considers the shutdown as a failure and leaves the resource in its previous state.

Batch Program Environment variable properties

Table 93: Environment variable properties

Property	Description
User	(Optional) The name or ID of the user account on the machine where the process is to be executed. This field allows you to determine the permissions that the process must have when executed.
Group	(Optional) (UNIX only) The name of the group.
Password	(Optional) The user account's password for authentication. It is stored in the configuration database.
Environment variables	(Optional) The environment variables specified by the user. These variables are provided during the execution of the Startup, Status, and Shutdown scripts.
Debug program	Debugs the Batch Program resource and includes debugging information in its output. This enables the jvmstart tool to write a trace log, and has no effect on the Startup, Status, or Shutdown scripts.

Batch Program default actions

OpenEdge Management supports log and email actions for Batch Program resource. You can create your own log and email actions, or use the sample email template for **Batch_Program_Stopped** action.

When a log file action is triggered, OpenEdge Management writes the start time, stop time, and status of the current Batch Program process to the configured log file.

When an email action is triggered, OpenEdge Management invokes the email action with a set of values appropriate for the process. The following table describes the properties used in the email message template:

Table 94: Email message template properties

Property	Description
HOSTLINK	The URL to OpenEdge management.
PIDFILE	The PID file configured for the Batch Program process.
PROCESSCONTAINER	The container name of the Batch Program resource.
PROCESSENVVARS	The environment variables configured for the Batch Program.
PROCESSPID	The PID of the last known process.
PROCESSSTARTTIME	The start time of the process, if available. This requires a PID that OpenEdge Management can query.
PROCESSSTATE	The reported state of the process such as initializing, starting, running, stopping, not running.
PROCESSSTATUSMSG	The Status message of the current process when the email action is triggered.
PROCESSSTOPTIME	The stop time of the process, if OpenEdge Management is able to determine it. Otherwise, the stop time of the end of the shutdown script.
PROGRAMNAME	The name of the Batch Program resource.
SHUTDOWNSCRIPT	The Shutdown script, if configured.
SHUTDOWNSCRIPTARGS	The Shutdown script arguments, if configured.
STARTUPSCRIPT	The Startup program, if configured.
STARTUPSCRIPTARGS	The arguments for the Startup program.
STATUSSCRIPT	The Status script, if configured.
STATUSSCRIPTARGS	The Status script arguments, if configured.
STDERRFILE	The stderr file configured for the Batch Program process.
STDOUTFILE	The stdout file configured for the Batch Program process.

Few properties mentioned in the table are not applicable when an email action is fired for a status failure without a process involved in it. For example, the properties PROCESSSTARTTIME, PROCESSSTOPTIME, PROCESSSTATE, PROCESSID are not applicable without a process object.

PID registration

To register a Batch Program for tracking using PID registration:

1. From the OpenEdge Management console menu, click **Resources** > **Go to Resources**.

   All the resources managed by your console appear in a grid frame.
2. Filter or search for, and select the required Batch Program resource.

   The **Batch Program: <instance name>** page appears.
3. In the **Command and Control** section, click **Configuration**.

   The **Edit Batch Program** page appears.
4. Click **EDIT**.
5. In the **Monitoring** section, select the **Monitoring method** as **PID registration**.
6. Select **Status failure behavior** as **Fire alert**.
7. In the following `curl` example, replace the Process ID in the URL and the PID in the JSON body as required and run it.

   ```
   curl -u admin:admin -H "Content-Type: application/json" -d '{"pid": 5380 }' -X POST
   http://localhost:9090/oem/containers/localhost/batch/localhost%3Aresource.batch.apoint/processes/5380
   ```

   The registered process appears in the **Processes** section of the **Batch Program: <instance name>** page.
8. In the **Shutdown** section, select **Shutdown method** as **Terminate** or **Shutdown script** to stop the process.
9. Click **SAVE**.

OpenEdge Management tracks a Batch Program's PID periodically based on the monitoring plan and notifies you when the process is no longer running by firing alerts, and runs the action associated with the default action.

Note: OpenEdge Management is unable to detect the exit code for a process managed through PID registration. In such case, if the process is stopped outside the control of OpenEdge Management, OpenEdge Management can only use the action associated with the default exit code. If you need OpenEdge Management to respond to specific exit codes, you must use a startup script and start the process with OpenEdge Management.

API references

The following API references enable you to manage a Batch program process:

- GET( ) method (Process) on page 207
- POST( ) method (Process) on page 208
- POST( ) method on page 209
- PUT( ) method (Process) on page 209
- DELETE( ) method (Process) on page 210
- GET( ) method (Processes) on page 211

GET( ) method (Process)

```
GET /containers/{container}/batch/{key}/processes/{processid}
```

Retrieves details of a specific process associated with the Batch Program.

A Batch Program configured for monitoring using a PID file, PID registration, or status script may have a process that has never started directly from OpenEdge Management. This method is used to track the instance when not using a startup script from within OpenEdge Management.

Endpoint URL

```
/containers/{container}/batch/{key}/processes/{processid}
```

- `{container}` refers to the name of the AdminServer on which the Batch Program process is running.
- `{key}` refers to the OpenEdge Management resource key for the Batch program process.
- `{processid}` refers to the Process ID assigned for tracking the Batch program process through the REST API.

HTTP method

GET

Parameters

None

Properties

Property	Description
`url`	The URL to access the details of the Batch Program process.

Sample request

None

Sample response

```
{name: "prowin2",
pid: 19280,
startTime: 1488325450000,
processId: "BCKZKSpvSMiZOy5BztrcqQ",
state: "Not Running",
statusMessage: "Not Running",
icon16: "/images/batch16.png",
icon32: "/images/batch32.png",
url:
"/oem/containers/localhost/batch/localhost%3Aresource.batch.prowin/processes/BCKZKSpvSMiZOy5BztrcqQ"

}
```

POST( ) method (Process)

```
POST /containers/{container}/batch/{key}/processes/{processid}
```

Registers a process that is to be tracked by the Batch Program using the specified ID.

For a Batch Program resource that is configured for PID registration, the PID must be available on the same machine as the AdminServer where the Batch Program resource is configured. You can provide your own specified identifier that is used for the PID.

Endpoint URL

```
/containers/{container}/batch/{key}/processes/{processid}
```

- `{container}` refers to the name of the AdminServer on which the Batch Program process is running.
- `{key}` refers to the OpenEdge Management resource key for the Batch program process.
- `{processid}` refers to the Process ID assigned for tracking the Batch program process through the REST API.

HTTP method
POST

Parameters
None

Sample request

```
{
pid: 19280
}
```

Sample response

```
{name: "prowin2",
pid: 19280,
```

```
startTime: 1488325450000,
processId: "19280",
state: "Not Running",
statusMessage: "Not Running",
icon16: "/images/batch16.png",
icon32: "/images/batch32.png",
url: "/oem/containers/localhost/batch/localhost%3Aresource.batch.prowin/processes/19280"
}
```

Code example
The following is a `curl` example to register a Batch Program for tracking using PID registration:

```
curl -u admin:admin -H "Content-Type: application/json" -d '{"pid": 5380 }' -X POST
http://localhost:9090/oem/containers/localhost/batch/localhost%3Aresource.batch.apoint/processes/1928
```

POST( ) method

`POST /containers/{container}/batch/{key}/processes`

This method is for the Batch Program process that is not assigned with a Process ID from the client in the URL. OpenEdge Management assigns a Process ID for this process.

Endpoint URL

```
/containers/{container}/batch/{key}/processes
```

- `{container}` refers to the name of the AdminServer on which the Batch Program process is running.
- `{key}` refers to the OpenEdge Management resource key for the Batch program process.

HTTP method
POST

Parameters
None

Sample request
None

Sample response
None

PUT( ) method (Process)

`PUT /containers/{container}/batch/{key}/processes/{processid}`

Updates the process to use a different PID.

Endpoint URL

```
/containers/{container}/batch/{key}/processes/{processid}
```

- `{container}` refers to the name of the AdminServer on which the Batch Program process is running.
- `{key}` refers to the OpenEdge Management resource key for the Batch program process.
- `{processid}` refers to the Process ID assigned for tracking the Batch program process through the REST API.

HTTP method
PUT

Parameters
None

Sample request

```
{
pid: 12334
}
```

Sample response
None

DELETE( ) method (Process)

```
DELETE /containers/{container}/batch/{key}/processes/{processid}
```

Removes the tracking of the specified Batch Program process and stops the process before removing it.

Note: Stopping the process before removing the process is optional.

The **Shutdown method** to stop the process can be either **Terminate** or **Shutdown script**. If this method stops the process before removing it, the process is no longer tracked. If this method is not specified, the tracking of the process is removed without stopping the process.

Endpoint URL

```
/containers/{container}/batch/{key}/processes/{processid}
```

- `{container}` refers to the name of the AdminServer on which the Batch Program process is running.
- `{key}` refers to the OpenEdge Management resource key for the Batch program process.
- `{processid}` refers to the Process ID assigned for tracking the Batch program process through the REST API.

HTTP method
DELETE

Parameters
This method uses `stop` as a parameter in the endpoint URL.

```
/containers/{container}/batch/{key}/processes/{processid}?stop=true
```

Parameter	Data type	Optional	Description
`stop`	boolean	Yes	Indicates if the process must be stopped before removing it. The default is set to `false`. If the shutdown of the process fails, the process tracking is not removed.

Sample request
None

Sample response
None

GET( ) method (Processes)

```
GET /containers/{container}/batch/{key}/processes
```

Retrieves a list of processes tracked by OpenEdge Management for a Batch Program.

Endpoint URL

```
/containers/{container}/batch/{key}/processes
```

- `{container}` refers to the name of the AdminServer on which the Batch Program process is running.
- `{key}` refers to the OpenEdge Management resource key for the Batch program process.

HTTP method
GET

Parameters
None

Sample request
None

Sample response
None

10

Configure third-party Web applications

The following topics describe how OpenEdge Management enables you to host and manage custom third-party Web applications available in your production environment.

For details, see the following topics:

- Overview
- Requirements for hosting a third-party Web application
- Host a third-party Web application
- Page contributions for a third-party Web application
- OpenEdge Management Extension framework
- Third-party Web application layout
- OpenEdge Management REST API framework
- Sample Web application in OpenEdge Management

Overview

OpenEdge Management helps you to monitor the availability and performance of OpenEdge resources. Furthermore, to accommodate management capabilities for Web applications beyond OpenEdge environment, OpenEdge Management also supports custom third-party Web applications.

This allows you to manage Web applications that are available in your own production environment through the OpenEdge Management console. In other words, you can now manage both OpenEdge resources as well as non-OpenEdge Web applications using a single management console.

With support for third-party Web applications, you can host your own Java Web applications in the OpenEdge Management Web server, integrate your own web pages, and extend existing OpenEdge Management functionality. Third-party Web applications can take advantage of OpenEdge Management's Web server security and styling, and specific extension points provided in OpenEdge Management.

Extension mechanism in OpenEdge Management

The extension mechanism in OpenEdge Management to support third-party Web applications allows you to:

- Store and retrieve resource properties.
- Host third-party standard Java Web applications within the Jetty Web server.
- Add your own web pages and access them through the management console main menu.
- Implement an extension point to contribute to your own landing page.
- Use the sample Web application shipped with OpenEdge Management as a template to take advantage of OpenEdge Management security and styling.
- Use the REST API framework implementation to design your own REST APIs within your Web application.

To begin with, the following are provided:

- A Web application template as a sample for resource implementation.
- Limited access to OpenEdge Management internals to perform the above mentioned tasks through restricted API.
- Documentation to access a custom Web application with custom resources and web pages in OpenEdge Management.

The following are not provided:

- Direct access to cache database or to create of new charts.
- Direct access to the OpenEdge Management configuration database.
- Direct access to OpenEdge Management class internals beyond what is available in the documentation.
- Direct access to remote AdminServers or to create new remote AdminServer plugins.

OpenEdge Management security

As you manage your Web applications using OpenEdge Management, one of the primary concerns is the security of the data transferred over the Internet. Attackers can exploit the vulnerabilities in Web applications to maliciously gain access to the management console and additional privileges to cause service disruption.

For security reasons, OpenEdge Management does not recommend using inline JavaScript. OpenEdge Management is configured to disable inline JavaScript when a web page request is made. And to address Cross-site scripting (XSS) attacks, OpenEdge Management recommends the use of penetration tools such as OWASP Zap to perform security testing for third party Web applications.

To simplify the security configuration, OpenEdge Management provides a SSO process for all Web applications running in its Web server. For more information, see Implement single sign-on (SSO) on page 215.

Requirements for hosting a third-party Web application

To host a third-party Web application in OpenEdge Management Web server and make the integration seamless, the following are required:

- **Single sign-on (SSO)** — Creating a single sign-on session between OpenEdge Management and your Web application to provide a single point of entry to users.
- **OpenEdge Management JSP tags** — Accessing OpenEdge Management's standard JSP tags allows you to share and create a consistent look and feel across the management console.
- **Web application support** — Hosting a standard Java Web application that consists of a web.xml file, JSP pages, images, style sheets, and JAR files.

Host a third-party Web application

OpenEdge Management is configured to use the Apache Tomcat that is capable of loading third-party Web applications. In OpenEdge Management, the Web user interface is a Web application that consists of JSP files, CSS files, and image files, and the associated Java code is hosted in the `fathom.jar` file. The OpenEdge Management Web application is considered the `ROOT` Web application. Third-party Web applications can have their own names without conflicting with the directory structure of the `ROOT` Web application.

You can host standard Web applications, for example `web.xml` and its related configuration items, that are supported by Tomcat. Third-party Web applications are hosted in the `$OEM/webapps` directory. You can edit the `fathomWebApplicationDir` system property in the `catalina.properties` file to change the directory and its default value.

Note: To begin with, you can use the sample Web application WAR file provided with the OpenEdge Management installation.

Though, by default, third-party applications have access to fathom internals, direct access to all internal methods is restricted. However, a set of public objects that contain a subset of OpenEdge Management internals are provided so that a third-party Web application can compile against these public objects.

Note: Progress reserves the right to modify any internal methods unless the methods are explicitly mentioned in OpenEdge documentation that they are available for third-party Web application use.

Implement single sign-on (SSO)

In general, Web applications hosted in a container are independent of other Web applications, and each Web application is configured with its own security implementation. To simplify the security configuration for third-party Web applications and provide a single point of entry to users, OpenEdge Management provides an SSO process—sharing a session cache implementation—for all Web applications running in its Web server.

To share a session cache implementation, you can use the sample Web application shipped with OpenEdge Management as a template. A custom session cache configuration shares session login information with the third-party Web application. You can use this configuration as an SSO solution for your third-party Web applications or provide your own security implementation.

Note: OpenEdge Management login configuration is available in the `web.xml` file and the `shiro.in` file in the sample Web application template that is shipped with OpenEdge Management.

Access JSP Standard Tag Libraries

To share and create a consistent look and feel between OpenEdge Management and third-party Web applications, you must access OpenEdge Management's standard JSP tags. These include:

- Toolbars, web page sections such as headers and footers, common style sheets, and common JavaScript libraries that are shipped with OpenEdge Management.

  You can see the web pages available in the sample Web application shipped with OpenEdge Management as an example. The common fathom style sheet is available in `/stylesheets/fathom.css`.

- JSP Standard Tag Libraries (based on the standard libraries in the Jetty package).

  Standard tag libraries are available in the sample Web application. However, if required, you can add other standard tag libraries for your application.

Note: OpenEdge Management ships with a full implementation of ExtJS and Kendo UI Professional frameworks. To use Kendo or third-party libraries, ensure that you have correct development licenses. Progress is not responsible for renewing the licenses purchased for these libraries that are used for third-party Web applications. Alternatively, you can use your own JavaScript libraries, but ensure that necessary library resources are included as part of your Web application.

Load Java classes

A class loader provides libraries and JSP pages, which are compiled at runtime into Java classes, to a Web application. The class loader of your Web application requires access to JAR files, a few internal classes, and any API classes of OpenEdge Management that must be exposed to third-party Web applications.

OpenEdge Management must ensure that the class loader is configured properly so that the class loader is created when the Web server is being initialized at the startup of OpenEdge Management. Each class loader of a third-party Web application is considered a child of OpenEdge class loader and remains independent of other Web applications. Each third-party Web application is defined as an extension to OpenEdge Management and the search path of the Web application class loader is allowed to access OpenEdge Management classes.

OpenEdge Management also provides shared libraries that allow third-party Web applications to host their own JAR files. The `$OEM/jars` directory acts as a shared library for all JAR files that are shipped with OpenEdge Management. The third-party Web applications can host their own JAR files in the `WEB-INF/lib` directory of the Web application.

Note: Each Web application must have its own class loader in order to work with the Jetty Web server.

Page contributions for a third-party Web application

OpenEdge Management provides hooks, located in predefined locations, which allow a third-party Web application to add user interface elements to every page. In this release, only a limited set of hooks are provided, for example, main menu and footers in the OpenEdge Management console.

To register these hooks and to inject a custom `PageContributor` class into OpenEdge Management, a registration mechanism is provided for third-party Web applications. During execution of every page, the methods in the page contributor instance are invoked at specific points.

Note: A third-party Web application must implement its own `PageContributor` class in order to work with OpenEdge Management.

OpenEdge Management implements specific callbacks for the page contributor to the following locations in the management console:

- **Page header** — To inject script and style.
- **Main menu** — To add a functional menu to the main menu bar.
- **Footer menu** — To add new entries to the footer.
- **Options page** — To add new entries to the **Options** page.

The extension mechanism provides the following configuration entry points to register extensions in OpenEdge Management:

- Extension name — The name of the extension.
- Extension class path — Extends the default behavior of OpenEdge Management.
- Extension initializer class — For startup, state, and shutdown of the extension.
- Standard file layout — The location of the configuration file of the extension.
- Standard Java interfaces — Supports the above entry points.
- Dashboard entry point class — Adds top-level pages.
- Configuration entry point class — Adds entries to the **Options** page in the management console.
- User login entry point page — Allow user logins to open the Dashboard page or any other page defined by the plugin.

Behavior of OpenEdge Management

OpenEdge Management implements callbacks to the exact contributor whenever necessary, and recognizes the specific locations in the management console as unique identifiers. These identifiers are hard-coded and can be used by menu contributors to determine the context. OpenEdge Management determines when to call the Contributor class, and may invoke the methods of a contributor more than once per page or may not invoke them at all.

For example, a callback for contributors to the OpenEdge Management console main menu receives a parent main `MENU-ITEM` object and a page context. The Contributor class adds child objects. In cases where the same callback is implemented to many locations, OpenEdge Management provides sufficient context information to the contributors to generate the required child objects.

OpenEdge Management ensures that each page element is uniquely identified. For example, for a parent `MENU-ITEM` that has been passed to the Contributor instance, OpenEdge Management sets a unique `"id"` property to the `MENU-ITEM` so that it can be uniquely identified.

Register a contributor

Third-party Web application menu contributor objects must be registered with OpenEdge Management. A contributor is registered when the third-party Web application is initialized. The third-party Web application provides a configuration property that defines a properties file that has the name of the Contributor class.

OpenEdge Management creates and implements the `ServletContextListener` class to ensure that the Contributor class is instantiated in the required Web application context and registered with OpenEdge Management. When registering the contributor class is no longer required, the `ServletContextListener` implementation ensures that the contributor is removed from memory.

The context listener loads the properties from the `WEB-INF/fathomextension.properties` file. This file contains details such as name, version, and properties that specify the contributor class. For more information, see the sample Web application.

OpenEdge Management Extension framework

OpenEdge Management allows extensions to contribute to the user interface to third-party Web applications. Predefined points are provided where the extensions can be added to the user interface. The primary points are main menu, page footers, and the **Options** page of the management console.

To accomplish this, a framework has been built to allow simple extensions through contributor class objects. Each individual point, where extensions are added to the user interface, requests necessary data using the fathom extension contributor classes. You can add other contributor classes to the framework, but only without altering the extension code of a previous version of the framework.

The extension framework allows hosting of third-party Web applications on the same Web server as that of OpenEdge Management. This enables OpenEdge Management to define the contributor classes as needed. The lifecycle of the extension points can be controlled using the lifecycle of the third-party Web application.

Access OpenEdge Management internals

Typically, OpenEdge Management is a Web application designed to be isolated from other Web applications. The extension framework restricts direct access to OpenEdge Management internals.

To allow third-party Web applications to access OpenEdge Management internals, a standard Java Web application servlet listener is written. This servlet listener searches for and loads necessary configuration files that have required contributor classes. This avoids hosting of third-party Web application JAR files in the OpenEdge Management Web application without restricting extensions to OpenEdge Management.

Third-party Web application layout

OpenEdge Management uses the Jetty Web server, a Java servlet container, which can host multiple Web applications. This allows the Web applications to load their own context listener. OpenEdge Management has single servlet context listener to apply extensions to the user interface. This context listener looks for a standard java properties file named `WEB-INF/fathomextension.properties` that has the configuration information for extensions to OpenEdge Management.

You can define extensions to OpenEdge Management by implementing specific classes and referencing known property names in the `fathomextension.properties` file to define which classes implement which interfaces.

An example of a servlet context listener specified in a `web.xml` file is provided in Table 1:

Table 95: WEB-INF/web.xml

```
<listener>
    <listener-class>com.progress.fathom.extension.ExtensionLoaderListener</listener-class>
</listener>
```

A sample `fathomextension.properties` file with a fathom extension initializer implementation and menu contributor implementation is provided in Table 2:

Table 96: WEB-INF/fathomextension.properties

```
# this is a sample fathom extension definition file.  This file contains
an example set of contributor classes that add UI elements to OpenEdge Management
# this is optional internal name of web extension
name=samplewebapp
# this is optional internal version of web extension
version=1.0
# this is an optional property that references a class that is an instanceof
IExtensionInitializer
initializer=com.progress.fathom.sample.webapp.SampleApplicationInitializer
# this is an optional property that references a class that is an instance of
IMenuContributor
menuContributor=com.progress.fathom.sample.webapp.SampleMenuContributor
# sample landing page contributor. The default fathom implementation does not do anything
landingPageContributor=com.progress.fathom.extension.preferences.FathomLandingPageContributor
```

When initialized, the servlet context listener instantiates the classes defined for known properties in the `fathomextension.properties` file. Though other properties in the file are ignored, they can be accessed from the initializer class implementation when required.

Once the extensions are loaded, they can be seen in the extension diagnostic page: `http://localhost:9090/diagnostics/extensions.jsp`.

Currently there are limited set of known properties that the context listener looks for, and they are described below:

Property name	Description
name	(Optional) Name of the fathom extension. This can be any string.
version	(Optional) Version of the fathom extension, displayed only in the extension diagnostic page. This can be any string.
initializer	(Optional) Fully qualified name of a class that implements `IExtensionInitializer`. A single instance of this class is stored in the memory until the Web application is terminated. This class can initialize any information required by the fathom extension.
menuContributor	(Optional) Fully qualified name of a class that implements `IMenuContributor`. A single instance of this class is stored in the memory until the Web application is terminated. This class is used by the fathom extension to add items to the main menu and footer menu of the console.
landingPageContributor	(Optional) Fully qualified name of a class that implements `ILandingPageContributor`. A single instance of this class is stored in the memory until the Web application is terminated. This class is used by the fathom extension to add landing pages that the user can select to use immediately after logging into the console.

OpenEdge Management REST API framework

The REST API in OpenEdge Management is built on an annotation-based framework to define a REST interface. HTTP clients use the REST interface to interact with OpenEdge Management. Third-party Web applications looking to extend OpenEdge Management functionality in their own Web application can use the OpenEdge Management REST API framework.

The REST API framework provides permission checks, parameterized URL matching, bi-directional object conversion of JSON-mapped entities, and handling upload of multi-part encoded files. The framework supports `GET`, `PUT`, `POST`, and `DELETE` HTTP methods. The sample Web application shipped with OpenEdge Management is an example of a simple REST API with sample `GET` and `POST` operations.

Note: The REST API framework in OpenEdge Management relies on the Jackson third-party library, an open source library available for use within your Web application.

Configure a third-party Web application

The REST API framework in OpenEdge is built on the standard JAVA servlet framework. The code for the REST API framework is built into the `fathom.jar` file and is available to third-party Web applications. The servlet receives its configuration from an XML file which specifies the classes that are to be scanned for annotations.

Note: The servlet must be defined in the Java Web application `web.xml` file.

An example of REST servlet definition and configuration in an XML file is provided in Table 3:

Table 97: WEB-INF/web.xml

```
<servlet>
    <servlet-name>RestServlet</servlet-name>
    <servlet-class>com.progress.fathom.FathomAPIServlet</servlet-class>
    <init-param>
        <param-name>configuration</param-name>
        <param-value>/WEB-INF/restconfig.xml</param-value>
    </init-param>
            <init-param>
        <param-name>urlprefix</param-name>
        <param-value>/rest</param-value>
    </init-param>

    <multipart-config>
    </multipart-config>
</servlet>
<servlet-mapping>
    <servlet-name>RestServlet</servlet-name>
    <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

The file format of the configuration XML file is simple. The file can define any number of implementations. Each `<fathomapi/>` node defines the fully qualified name of a class that extends the `FathomAPIServletHandler` class. To handle URL mapping and request access to a context, the derived class must inherit from the `FathomAPIServletHandler` class.

The XML file specifies the classes that are to be scanned for annotations to map them to URLs. Servlet mapping is defined in the web.xml file and all the URLs in the classes are made available for mapping by the Java servlet beneath this servlet mapping. In the above example, all the URLs mapped from the classes are made available beneath `/<webcontextpath/rest/`.

An example of a complete `restconfig.xml` file is provided in Table 4:

Table 98: WEB-INF/restconfig.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<fathomapis>
    <fathomapi>
        <!--  this is optional. This provides standard /api human readable content -->
        <class>com.progress.fathom.api.OEMAPI</class>
    </fathomapi>
    <fathomapi>
        <!--  simple example that shows how to consume and generate JSON content via
REST API -->
        <class>com.progress.fathom.sample.webapp.RestExample</class>
    </fathomapi>
</fathomapis>
```

REST API code example

An example of the code used in the `samplwebapp.war` file shipped with OpenEdge Management is provided in Table 5:

Table 99: RestExample.java

```
package com.progress.fathom.sample.webapp;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.ObjectNode;
import com.progress.fathom.Body;
import com.progress.fathom.Description;
import com.progress.fathom.FathomAPIServletHandler;
import com.progress.fathom.PathParam;
import com.progress.fathom.QueryParam;
import com.progress.fathom.RequestException;
import com.progress.fathom.RequestMapping;
import com.progress.fathom.RequestMethod;
import com.progress.fathom.RequiredResourcePermission;
import com.progress.fathom.Since;
import com.progress.isq.ipqos.resources.security.SecurityBean;
/**
 * this is sample REST API handler based on the OpenEdge Management REST
 * annotation mapping
 *
 */
public class RestExample extends FathomAPIServletHandler {
    @Description("Example REST get request")
    @RequestMapping(path = "/get/{id}", method = RequestMethod.GET)
    @Since("11.7")
    @RequiredResourcePermission(grantName = SecurityBean.SECURITY_ROOT, action =
SecurityBean.ACTION_READ)
    public Object getSample(@PathParam("id") String id, @QueryParam("q") String
queryString) throws RequestException {
        // object mapper is from jackson JSON library
        ObjectMapper mapper = new ObjectMapper();
        // this creates a simple JSON response using the values from the URL
        // mapped back into the object as JSON object properties
        ObjectNode response = mapper.createObjectNode();
        response.put("id", id);
        return response;
    }
    @Description("Example REST POST request")
    @RequestMapping(path = "/post/{id}", method = RequestMethod.POST)
    @Since("11.7")
    @RequiredResourcePermission(grantName = SecurityBean.SECURITY_ROOT, action =
SecurityBean.ACTION_READ)
    public Object postSample(@PathParam("id") String id, @Body ObjectNode body) throws
 RequestException {
        // object mapper is from jackson JSON library
        ObjectMapper mapper = new ObjectMapper();
        // this creates a simple JSON response using the values from the URL
        // mapped back into the object as JSON object properties, along with the
        // content of the request body
        // from the POST
        ObjectNode response = mapper.createObjectNode();
        response.put("id", id);
        response.put("request", body);
        return response;
    }
}
```

The REST API framework scans each class for methods that are public and have a `RestMapping` annotation defined. The annotation specifies the URI pattern and HTTP method that are to be handled. The `RequestMapping` annotation with a specified path parameter is the least required annotation. If the method parameter is not specified, the annotation considers HTTP `GET` as the default method parameter.

Note: Each class can contain as many annotated methods as required. Each method must handle only one combination of URI pattern and HTTP method.

URL mapping

The REST API framework performs URL mapping based on the path specified in the `RequestMapping` annotation. The URL mapping accepts parameters defined within the URL string, and then the mappings are made available to parameters for a method through annotations.

The REST API code example on page 223 shows both URL and query-parameter mapping. The mapping is based on a simple pattern matching of the incoming URL, and each parameter is treated as a wildcard string for its position within the URL. Only the URL part of the request is matched against the annotated REST handler methods, and the query string of the URL is ignored.

Consider an example where the path: `/some/{container}/url/{key}` has two parameters defined in it: `container` and `key`. Their values from the URL can be retrieved either from annotated method parameters or from within the body of the method. In the REST API code example on page 223, the parameters to the method are annotated with `PathParam("<url parameter>")`, where the `url parameter` is the name matching the value in the path for the method.

Exception handling

The REST API framework expects the REST handler methods to throw exceptions of `RequestException` type. The `RequestException` requires an HTTP response code and a description text that is used to return an error to the client. However, an inner exception may also be passed to the constructor.

If a `RequestException` is thrown from a REST handler method, the framework considers it as a known exception and returns a JSON formatted response body with error text to the client. In some cases, the REST handler method might throw any subclass of `RequestException`.

The framework considers the `RequestException` as a normal termination of the REST handler method and does not log the exception. On other hand, an exception that does not extend `RequestException` is considered as an application error and recorded in the fathom error log file. The client then receives a response body with error text `HTTP 500 Internal Server Error` from the exception.

An example of `RequestException`:

```
@RequestMapping(path = "/get", method = RequestMethod.GET)
public Object getSample(@QueryParam("id") String idFromQuery) {
    if ("0".equals(id)){
        throw new RequestException("Invalid id", HttpServletResponse.SC_BAD_REQUEST);
    }
    ...
}
```

The corresponding response:

```
{
    error: {
        stack: [ ],
        number: "403",
        text: "Invalid id",
        source: "OpenEdge Management"
    }
}
```

REST handler method response

Each REST handler method must send a response or throw an exception. The method may handle the response or allow the framework to convert the method return value to a JSON string. All responses are can be Content-Type specified as application/JSON. Though this is not mandatory, the handler method must handle the HTTP Content-Type header and response body.

The response from a method can be one of the following types:

- **String** — If the type of object returned from the method is String, it is returned as-is with no validity checks. The method must ensure that the string is a properly formatted JSON encoded string.
- **ObjectNode** — If the type of object returned from the method is ObjectNode, a part of Jackson JSON library, the framework serializes it and writes it back to the client.
- **Plain old Java object** — If the type of object returned from the method is a Java object, the framework attempts to convert it to a JSON encoded string using the built-in Jackson library. If the object cannot be converted to a JSON encoded string, an `HTTP 500 Internal Server Error` is returned to the client, and the exception is recorded in the fathom error log file.

Note: Returning a string or an ObjectNode is quicker than returning a Java object because the latter avoids using reflection to serialize the object.

For details about Jackson capabilities, see https://github.com/FasterXML/jackson.

An example of a method returning String:

```
@RequestMapping(path = "/get", method = RequestMethod.GET)
public Object getSample(@QueryParam("id") String idFromQuery) {
    return "{ \"id\" : \"someid\"}";
}
```

An example of a method returning ObjectNode:

```
@RequestMapping(path = "/get", method = RequestMethod.GET)
public Object getSample(@QueryParam("id") String idFromQuery) {
    ObjectMapper mapper = new ObjectMapper();
    ObjectNode resp = mapper.createObjectNode();
    resp.put("id", "someid");

    return resp;
}
```

An example of a method returning Plain old Java object:

```
@RequestMapping(path = "/get", method = RequestMethod.GET)
public Object getSample(@QueryParam("id") String idFromQuery) {

    MyThing thing = new MyThing();
    thing.setId("someid");

    return thing;
}
```

Method annotations

Several annotations are available at method level that perform method-related tasks. A few are provided here:

`Description`

Accepts a text string as its value. The text appears in the API method description displayed from `"/api"` request handler.

Note: This annotation is for documentation purposes only.

If the `com.progress.fathom.api.OEMAPI` class is listed in the `restconfig.xml` file, the `"/api"` URL is mapped and provides a human-readable description of the URLs that are mapped for the Web application. The description text is included in the response.

Refer to the REST API for the sample Web application on page 231 for an example of an API for a sample Web application containing description text.

`Since`

Accepts a text string as its value. Though this text does not appear in any example, you can use it as you see fit.

Note: This annotation is for documentation purposes only.

`RequiredResourcePermission`

Adds an authorization check performed by the fathom REST servlet implementation. Authorization checks in OpenEdge Management are based on a security object identified (a string), and an action.

If the user making the REST request exists and is authenticated, the authentication check must succeed to invoke the method. If the user is not authenticated, the REST request fails with a `403 Forbidden` response. If the user exists and the `RequiredResourcePermission` annotation is not available, the user is authorized to access OpenEdge Management to perform the API call.

At the least, the following annotation must be added to prevent users without normal resource permissions, such as the PSCTrend role, from accessing the method. However, users with PSCOper and PSCAdmin roles will still be able to access the method.

```
@RequiredResourcePermission(grantName = SecurityBean.SECURITY_ROOT, action =
SecurityBean.ACTION_READ)
```

For a list of predefined permission objects, see the `SecurityBean` class. At the least, your Web application should check `SecurityBean.SECURITY_RESOURCE,action=SecurityBean.ACTION_READ`.

Note: Three predefined user roles—PSCOper, PSCAdmin, and PSCTrend—are available in OpenEdge Management, and permissions are granted individually based on the user role. PSCOper and PSCAdmin user roles have full read permissions on security_resource. The PSCAdmin user role has full read/write permissions on all security permission tokens within OpenEdge Management. The PSCTrend user role has limited permissions to a few APIs that are required for remote trending.

Parameter annotations

Several parameter annotations are available that perform type conversion, resource lookup, URL parameter mapping, and query parameter mapping from the HTTP request. The `FathomAPIServletHandler` has methods to access the URL, query parameters, HTTP requests and response objects.

You can define any number of parameters for a method. If any parameter cannot be resolved based on its annotation, the parameter value is set to null for Object parameters and default value for Java scalar types (int, float).

A few annotations are provided for your convenience; using them is not mandatory:

`QueryParam`

Add this to a method parameter to retrieve a value from the query to the URL.

For example:

```
@RequestMapping(path = "/get", method = RequestMethod.GET)
public Object getSample(@QueryParam("id") String idFromQuery) {
    ...
}
```

`PathParam`

Add this to a method parameter to retrieve a value from a parameterized URL.

For example:

```
@RequestMapping(path = "/get/{id}", method = RequestMethod.GET)
public Object getSample(@PathParam("id") String id) {
    ...
}
```

`RequestPathParam`

Add this to a method parameter to assign the parameter to a source object based on a fully qualified resource key. The value of the key is derived from a URL parameter.

If the resource defined from the value of the key does not exist, the method is not called and a `404 Resource Not Found` response is returned to the caller. If the resource exists, but cannot be cast to the data type defined by the method parameter, the caller receives a `403 Bad Request` with a JSON response body. The error response indicates that incorrect resource type was discovered by the resource key in the URL.

For example:

```
@RequestMapping(path = "/get/{key}", method = RequestMethod.GET)
    public Object getResource(@ResourcePathParam("key") IResource resource) {
        ...
    }
```

`Files`

Handles an HTTP multi-form encoded request that contains files. The files are saved to disk and passed as a list of `UploadedFile` objects with details from the `POST` method.

For example:

```
@RequestMapping(path = "/files", method=RequestMethod.POST)
public Object postFiles(@Files List<UploadedFile> files) {
}
```

`Body`

Maps the body of the resource to the method parameter. The framework assigns the value based on the data type of the parameter. This is only for use with `PUT`, `POST`, and `DELETE` methods. HTTP requests that user `GET` method do not have body content.

Note: To ensure that the string conversion behaves properly, the client must send the character set information with the Content-Type header when posting the data using the `POST` method. If the charset field of the Content-Type header is not available, OpenEdge Management assumes that the character set is `IS8859-1`. If the Content-Type is specified as application/JSON, the header **must** be `UTF-8`.

An example with String as the data type is provided in Table 6:

Table 100: String REST API method

```
@RequestMapping(path = "/content", method=RequestMethod.POST)
public Object postFiles(@Body String requestBody) {
}
```

OpenEdge Management uses the Jackson JSON parser. If the request Content-Type is specified as application/JSON, OpenEdge Management attempts to read the request content and convert it to a Jackson ObjectNode implementation.

An example with Jackson ObjectNode as the data type in Table 7:

Table 101: Jackson ObjectNode REST API method

```
@RequestMapping(path = "/post/{id}", method = RequestMethod.POST)
public Object postSample(@PathParam("id") String id, @Body ObjectNode body) {
}
```

Jackson can convert a JSON structure parsed from a string to a Plain Old Java Object based on standard getter/setter and few annotations. It also supports some annotations on the entity class methods to determine how deserialization is to be performed. The OpenEdge Management REST API framework built this ability to REST handler methods via annotations.

The following example defines a class `MyThing` and uses `Body` annotation to import a request body from the `POST` method to a `MyThing` object. Jackson maps properties from the JSON request body.

Table 102: Sample JSON POST content

```
{
    "id" : "someid",
    "description" : "some description"
}
```

Table 103: MyThing.class and REST API method

```
@RequestMapping(path = "/post/{id}", method = RequestMethod.POST)
public Object postThing(@PathParam("id") String id, @Body MyThing thing) {

}

public class MyThing {
    private String id;
    private String description:

    public String getId() {
        return id;
    }

    public String getDescription() {
        return description;
    }

    public void setId(String id) {
        this.id = id;
    }

    public void setDescription(String description) {
        this.description = description;
    }
}
```

Sample Web application in OpenEdge Management

A typical install of OpenEdge Management comes with a sample Web application WAR file and its original source code bundled in a ZIP file included in the `$OEM/samples` directory. The sample Web application is a single page application with a sample REST API. To access the REST API sample, see Access the REST API extension on page 231. To begin with, you can use this sample Web application as a template for third-party Web applications.

The sample Web application provides the following:

- Java Web application definition including the `web.xml` file and the required file structure.
- Configuration for Web application security using Shiro. OpenEdge Management uses Shiro for its security configuration.
- A sample REST API implementation based on REST annotated classes in OpenEdge Management.
- Hooks to add menu entries to menus in OpenEdge Management.
- A sample web page with OpenEdge Management styling that can be used as a template for custom web pages.

Use the sample Web application

The sample Web application shipped with OpenEdge Management is available in the `$OEM/samples` directory.

To use the sample Web application:

1. Stop the AdminServer (using the command `fathom -stop` or `proadsv -stop`).
2. Navigate to OpenEdge Management install directory `$OEM` on your machine.

   Typically, this is the `C:\Progress\oemgmt` directory on your Windows machine.
3. Open the **Samples** folder to find:
   - A sample Web application `samplewebapp.war` file, which is a compiled and deployable Web application archive (WAR) file readily available for use.
   - The source file for the sample Web application `samplewebapp.src.zip` bundled in a ZIP file.
4. Copy the sample Web application `samplewebapp.war` file to the **webapps** folder in the OpenEdge Management install directory.

   If this folder does not exist, you must create a new folder named **webapps** in the OpenEdge Management install directory and then copy the WAR file into the folder.
5. Restart the AdminServer (using the command `fathom -start` or `proadsv -start`).
6. Open and log into the OpenEdge Management console using `http://localhost:9090`.

   After logging in, the OpenEdge Management console provides the following extensions as part of the sample Web application:
   - A functional menu, **Sample**, in the main menu bar for resource implementation.
   - A link in the **Options** page, **Sample Config**, to add your own preferences to the third-party Web application.

- A link in the footer, **Sample**, to provide access to your own web pages or online help.

The sample Web application defines a menu contributor that adds a new menu item, **Sample**, to the main menu and the footer in the management console. The new menu item is named as **Sample**; it has few example links beneath it.

A landing page contributor is also defined in the sample Web application. It includes a landing page item for the OpenEdge Management jobs home page. To choose a landing page example, select **Options** > **User Preferences** in the management console.

Access the REST API extension

To access the REST API extension for the sample Web application shipped with OpenEdge Management:

1. Enter the URL `http://localhost:9090/samplewebapp` in your web browser.

   The **Sample Web Application Example Page** appears in the OpenEdge Management console.
2. In the **GET example** section:
   1. Click **rest/get/id1** to see a sample API for `GET`.
   2. Click **rest/api** to see a list of REST APIs available for the sample Web application.
3. In the **POST example** section:
   1. Provide a value as input in the **Enter some value** field.
   2. Click **POST**.
   3. The value returned using the API for `POST` appears in the **Output shows up here** field.

REST API for the sample Web application

An example of a sample Web application REST API available in OpenEdge Management is provided in Table 10:

Table 104: /rest/api

```
Service URLs marked with a + are generic APIs where the URL handles several APIs.
-----------------------------------------------
  /rest/api

  GET:
    Description: returns a human readable list of API methods provided by fathom REST
interface

-----------------------------------------------
  /rest/get/{id}

  GET:
    Description: Example REST get request

-----------------------------------------------
  /rest/post/{id}

  POST:
    Description: Example REST POST request
```

Access resources

You can access both OpenEdge and non-OpenEdge resources using the **Sample Web Application Example Page** in OpenEdge Management.

To access the resources:

1. Enter the URL `http://localhost:9090/samplewebapp` in your web browser.

   The **Sample Web Application Example Page** appears in the OpenEdge Management console.

2. In the **Search for resource** field of the **Resources search example** section, type some text relevant to the name or description of the resource that you want to access.

   The name and description of first few relevant resources appear in a list.

3. From the list, select the required resource to access its home page.

Compile sample Web application using Ant

You can compile the sample Web application using an example `build.xml` file.

To compile the sample Web application:

1. Extract the `samplewebapp.src.zip` file available in the `$OEM/samples` directory.
2. Create a `build.xml` file in the top-level directory using the example file provided below.
3. Start Proenv.
4. Change the directory to the directory that contains the sample Web application code.
5. Run `proant`.

   The following is the example `build.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="samplewebapp" default="build" basedir=".">
    <!-- make environment variables available as properties -->
    <property environment="env"/>
    <!-- Compiler settings. -->
    <property name="javacFailOnError" value="true"/>
    <property name="javacDebugInfo" value="on"/>
    <property name="javacVerbose" value="false"/>
    <property name="logExtension" value=".log"/>
    <property name="compilerArg" value=""/>
    <property name="ant.build.javac.source" value="1.8"/>
    <property name="ant.build.javac.target" value="1.8"/>

    <!-- directory names for convenience -->
    <property name="DLC" value="${env.DLC}" />
    <property name="OEM" value="${env.OEM}" />
    <property name="PASOE" value="${DLC}/servers/pasoe" />
    <property name="build.src.dir" value="${basedir}/src" />
    <property name="build.out.dir" value="${basedir}/build" />
    <property name="build.class.dir" location="${build.out.dir}/classes" />
    <property name="samplewebapp.webcontent.dir" value="${basedir}/WebContent"/>
    <property name="samplewebapp.jar" location="${build.out.dir}/samplewebapp.jar"
/>
   <property name="samplewebapp.war" location="${build.out.dir}/samplewebapp.war"/>

    <property name="samplewebapp.src.zip"
location="${build.out.dir}/samplewebapp.src.zip"/>
```

```
    <!-- classpath for building samplewebapp.jar -->
    <path id="compile.classpath">

        <!-- include progress.jar -->
        <pathelement path="${DLC}/java/progress.jar" />
        <pathelement path="${DLC}/java/messages.jar" />

        <!-- include fathom.jar -->
        <pathelement path="${OEM}/jars/fathom.jar"/>

        <!-- java servlet specific stuff needed for most web pages -->
        <pathelement path="${PASOE}/lib/servlet-api-*.jar" />
        <!-- jackson JSON library for REST API-->
        <pathelement path="${OEM}/jars/jackson-core-*.jar" />
        <pathelement path="${OEM}/jars/jackson-annotations-*.jar" />
        <pathelement path="${OEM}/jars/jackson-databind-*.jar" />

        <!-- shiro for security -->
        <pathelement path="${OEM}/jars/shiro-core-*.jar"/>
        <pathelement path="${OEM}/jars/shiro-web-*.jar" />
    </path>
    <target name="build" depends="clean,compile,jar,war,zipsrc"
        description="Compile all classes and build jars for samplewebapp">
    </target>
    <target name="compile" depends="generic.java.compile"
        description="Compile all classes for samplewebapp.jar." >
    </target>

    <target name="generic.java.compile"
        description="Compile all classes in ${build.class.dir} as javasrc
${ant.build.javac.source}">
        <mkdir dir="${build.out.dir}" />
        <mkdir dir="${build.class.dir}" />
        <javac
            srcdir="${build.src.dir}"
            destdir="${build.class.dir}"
            includes="**/*.java"
            failonerror="${javacFailOnError}"
            verbose="${javacVerbose}"
            debug="${javacDebugInfo}"
            includeAntRuntime="no" >
            <!-- <compilerarg value="-Xlint"/> -->
            <classpath refid="compile.classpath" />
        </javac>
    </target>
    <tstamp>
        <format property="TODAY" pattern="yyyy-MM-dd HH:mm:ss" />
    </tstamp>
    <target name="jar" description="create samplewebapp.jar">
        <tstamp/>
        <jar destfile="${samplewebapp.jar}"
            manifest="${basedir}/META-INF/MANIFEST.MF" >
            <manifest>
                <attribute name="url" value="http://www.progress.com" />
                <attribute name="Vendor" value="Progress Software" />
                <attribute name="Bundle-Name" value="OpenEdge Management Sample Web
 Application" />
                <attribute name="Built-Date" value="${TODAY}"/>
            </manifest>
            <fileset dir="${DLC}">
                <include name="version" />
            </fileset>
            <fileset dir="${build.class.dir}">
                <include name="com/**/*.class" />
            </fileset>
            <!-- make sure to include any config files -->
            <fileset dir="${build.src.dir}" >
                <include name="com/**/*.xsl" />
```

```
                <include name="com/**/*.xml" />
                <include name="com/**/*.txt" />
                <include name="com/**/*.properties" />
            </fileset>
        </jar>
    </target>

    <target name="clean" description="clean output for sample web application">
        <delete dir="${build.class.dir}" />
        <delete file="${samplewebapp.jar}" />
        <delete file="${samplewebapp.war}" />
        <delete file="${samplewebapp.src.zip}" />
    </target>
    <target name="war" description="build samplewebapp .war file">
        <war
            destFile="${samplewebapp.war}"
            webxml="${samplewebapp.webcontent.dir}/WEB-INF/web.xml">

            <fileset dir="${samplewebapp.webcontent.dir}">
            </fileset>
            <lib dir="${build.out.dir}">
                <include name="samplewebapp.jar"/>
            </lib>
        </war>
    </target>
    <target name="zipsrc" description="create source samplewebapp.zip">
        <zip
            destfile="${samplewebapp.src.zip}"
            basedir="${basedir}"
            excludes="build.xml">
        </zip>
    </target>
</project>
```

OpenEdge release 11.7 ships with a copy of Ant build tool. A script called `proant` which sets up and runs the Ant is available in the `$DLC/bin` directory. Running `proant` uses the `build.xml` file to compile and generate the `samplewebapp.war` file into a directory named `build`. The example `build.xml` script can be used as it is or adjusted to your own environment. The `build.xml` file requires some environment variables that are normally set by Proenv.

The sample Web application also contains `.project` and `.setting` files for eclipse. You can import the sample Web application source code as an eclipse project and modify it as required. You can this project to create your own Web application.

Index

4GL debugger enabled property
  AppServer *113*
  WebSpeed Transaction Server *159*

A

Activate property *114*
AdminServer *13*
After image buffers property *27*
After image process property *27*
After image stall property *27*
After-image archival directory list property *27*
After-image archival interval property *28*
After-image Writer *45*
Agent maximum port property *49*
Agent minimum port property *49*
Agent name property *49*
Agent Shutdown Action property *50*
AIW *45*
Alternate buffer pool property *24*
Apache Tomcat *170*
Append to broker log file property
  AppServer broker *103*
  AppServer messaging *117*
  MS SQL Server DataServer *73*
  Oracle DataServer *61*
  WebSpeed broker *147*
  WebSpeed messaging *161*
Append to NameServer log file property *86*
Append to server log file property
  AppServer messaging *118*
  WebSpeed agent *155*
  WebSpeed messaging *162*
Application service names property
  AppServer *102*
  WebSpeed *146*
Application URL property *158*
AppServer Internet Adapters
  actions allowed on *127*
  configuration overview *126*
  creating instances *133*
  deleting instances *135*
  editing a configuration *133*
  enabling and disabling *135*
  properties
    advanced features *133*
    controlling NameServer *128*
    general *128*
    logging setting *129*
    TLS *132*
  viewing log files *134*
AppServer Keepalive property *133*
AppServers
  actions allowed on *98*
  additional management *126*
  agent properties
    advanced features *113*
    general *109–110*, *113*
    logging setting *110*
    pool range *113*
  agents
    adding or trimming *120*
  broker properties
    advanced features *105*
    AppService Name List *102*
    controlling NameServer *101–102*
    general *100–101*
    logging setting *102*
    owner information *101*
  configuration overview *97*
  creating instances *119*
  deleting instances *125*
  editing a configuration *119*
  environment variables *118*
  features *98*
  messaging properties *116–117*
  property storage *99*
  starting
    automatically *120*
    manually *119*
  status *121*
  TLS properties
    advanced features *116*
    general *115*
  viewing log files *126*
APW *45*
Assigning database default configurations *39*
Asynchronous Page Writer *45*
Asynchronous page writers property *26*
Authorized IP list property *142*
Auto start property
  AppServer *100*
  MS SQL Server DataServer *71*
  NameServer *85*
  OpenEdge database *22*
  Oracle DataServer *59*
  WebSpeed Transaction Server *145*
Auto trim timeout property
  AppServer *106*
  WebSpeed Transaction Server *150*
Automatically add After imaging extents property *53*

B

Background writers *45*
Backup method property *53*
Base index number property *36*
Base table number property *36*
Batch Program home *189*
Before image buffers property *26*
Before image cluster age property *27*
Before image delay writers property *27*
Before image process property *26*
Before image stall property *26*
Before image threshold property *26*
Before-image Writer *45*
Begin After imaging after transition property *53*
Binary upload max size property *158*
BIW *45*
Blocks in database buffers property *23*
Broker keep alive timeout property *85*
Broker log file threshold size property
- AppServer *105*
- MS SQL Server DataServer *73*
- Oracle DataServer *61*
- WebSpeed Transaction Server *149*

Broker log filename property
- AppServer *102*
- MS SQL Server DataServer *72*
- Oracle DataServer *61*
- WebSpeed Transaction Server *146*

Broker logging entry types property
- AppServer *103*
- MS SQL Server DataServer *73*
- Oracle DataServer *61*
- WebSpeed Transaction Server *147*

Broker logging level property
- AppServer broker *102*
- AppServer messaging *117*
- MS SQL Server DataServer *73*
- Oracle DataServer *61*
- WebSpeed broker *147*
- WebSpeed messaging *161*

Buffer hash latch factor property *24*

C

Case table name property *28*
Changing
- AppServer
  - agent properties *109*
  - broker properties *100*
  - environment variables *118*
  - messaging properties *116*
  - TLS properties *115*
- AppServer Internet Adapter properties *127*
- MS SQL Server DataServer
  - broker properties *70*

Changing *(continued)*
- MS SQL Server DataServer *(continued)*
  - environment variables *76*
  - server properties *74*
- NameServer
  - environment variables *90*
  - properties *83*
- OpenEdge database
  - basic properties *22*
  - default properties *23*
  - server group properties *40*
- Oracle DataServer
  - broker properties *58*
  - environment variables *64*
  - server properties *62*
- properties *174*
- WebSpeed Messenger
  - properties *138*
- WebSpeed Transaction Server
  - agent properties *153*
  - broker properties *144*
  - environment variables *162*
  - messaging properties *160*
  - TLS properties *159*

Changing OE Web Server run-time properties *178*
Client Type property *42*
Cluster enabled property *22*
Collation table property *28*
Configuration group *20*
Configuring
- AppServer components *97*
- AppServer Internet Adapters *126*
- AppServers *97*
- Batch Program *187*
- DataServers *57*
- MS SQL Server DataServers *69*
- NameServers *81*
- OpenEdge databases *19*
- Oracle DataServers *57*
- SonicMQ Adapters *167*, *169*
- Web service instances *180*
- WebSpeed components *137*
- WebSpeed Messengers *137*
- WebSpeed Transaction Servers *143*

Configuring Batch Program *188*
conmgr.properties file *20*
Connect property *114*
Connection Timeout property
- replication
  - agent *49*
  - control agent *52*

Controlling NameServer property
- AppServer *101*
- MS SQL Server DataServer *71*
- Oracle DataServer *60*
- WebSpeed Messenger *139*

Controlling NameServer property *(continued)*
WebSpeed Transaction Server *146*
Conversion map property *28*
CPU statistic *46*, *124*
Create after-image archival directories property *28*
Create Batch Program *188*
Creating
AppServer instances *119*
AppServer Internet Adapter instances *133*
MS SQL Server DataServer instances *76*
NameServer instances *91*
OpenEdge database
configurations *39*
server groups *43*
Oracle DataServer instances *65*
WebSpeed Transaction Server instances *163*
Critical agent property *52*

D

Data service name list property
MS SQL Server DataServer *72*
Oracle DataServer *60*
Database group *20*
Database name property *22*
Database property
replication
agent *49*
control agent *52*
server *50*
Database Role property *53*
Database service communication area size property *29*
DataServers *57*
Deactivate property *114*
Debug mode property *158*
Default configuration property *22*
Default cookie domain property *158*
Default cookie path property *158*
Defer Agent Startup property *50*
Deleting
AppServer instances *125*
AppServer Internet Adapter instances *135*
MS SQL Server DataServer instances *77*
NameServer instances *91*
OpenEdge database
configurations *47*
references *47*
server groups *48*
Oracle DataServer instances *65*
WebSpeed Transaction Server instances *165*
Deploying a REST application (.war file) *176*
Direct I/O property *25*
Disable client verification of TLS host name property *141*
Disable TLS session cache property *37*
Disable TLS session caching property *116*, *141*, *160*
Disable TLS session reuse property *141*
Disconnect property *114*

E

Editing
AppServer Internet Adapter properties *133*
AppServer properties *119*
MS SQL Server DataServer properties *76*
NameServer properties *90*
OpenEdge database
default properties *38*
server group properties *43*
Oracle DataServer properties *65*
WebSpeed Messenger properties *142*
WebSpeed Transaction Server properties *163*
Editing Batch Program *188*
Enable dynamic property updates property
WebSpeed Transaction Server *151*
Enable TLS client connections property
AppServer *115*
WebSpeed Messenger *141*
WebSpeed Transaction Server *159*
Enable TLS for remote connections property *37*
Environment property *89*
Execution Time Limit property *159*

F

Fault tolerance
connection-level *92*
server-level *93*
File upload directory property *158*
Full on-line and off-line backup arguments property *53*

G

Getting started *15*
Group name property
AppServer *101*
MS SQL Server DataServer *71*
Oracle DataServer *59*
WebSpeed Transaction Server *145*

H

Hash table entries property *24*
Host name property *84*
Host property
database server group *43*
replication control agent *52*

I

Incremental backup arguments property *53*
Index range size property *36*

Initial number of servers to start property
AppServer 113
WebSpeed Transaction Server 157
Internal administration command property 142

J

Java classpath property 36
jvmargs property
AppServer 107
WebSpeed Transaction Server 151

K

Keep Alive property 50
Kernel time statistic 47, 125
Key alias name property 37
Key alias password property 37

L

Load balancing
priority weight 93
Location property
NameServer 84
Location transparency 93
Lock table entries property 24
Lock table hash table size property 30
Log code page property 29
Log file threshold size property
NameServer 87
WebSpeed Messenger 141
Log file viewer
Batch Program 191
Log File Viewer (stderr) 191
Log File Viewer (stdout) 191
Log filename property
WebSpeed Messenger 140
Log files
AppServer 126
AppServer Internet Adapter 134
MS SQL Server DataServer 79
NameServer 96
OpenEdge database 21
Oracle DataServer 68
WebSpeed Messenger 142
Logging entry types property
NameServer 87
WebSpeed Messenger 140
Logging level property
WebSpeed Messenger 140

M

Managed databases 20
Managing Batch Program 187
Managing database instances 21
Max servers property 25
Max users property 24
Maximum client instances property
AppServer 105
MS SQL Server DataServer 73
Oracle DataServer 62
WebSpeed Transaction Server 149
Maximum clients per server property 42
Maximum dynamic port property 42
Maximum message property 52
Maximum NameServer client port property 139
Maximum number of broker log files property
AppServer 105
MS SQL Server DataServer 73
Oracle DataServer 61
WebSpeed Transaction Server 149
Maximum number of database areas property 30
Maximum number of JTA transactions property 37
Maximum number of log files property
NameServer 87
WebSpeed Messenger 141
Maximum number of server log files property
AppServer 113
WebSpeed Transaction Server 157
Maximum port number property
AppServer 110
MS SQL Server DataServer 75
Oracle DataServer 64
WebSpeed Transaction Server 154
Maximum private database buffers per user property 25
Maximum servers property
AppServer 113
WebSpeed Transaction Server 157
Maximum shared-memory segment size property 32
Message buffer size property 41
Messenger executable filename property 139
Minimum clients per server property 42
Minimum dynamic port property 43
Minimum NameServer client port property 139
Minimum port number property
AppServer 110
MS SQL Server DataServer 75
Oracle DataServer 64
WebSpeed Transaction Server 154
Minimum servers property
AppServer 113
WebSpeed Transaction Server 157
Modifying, See Changing
Modifying configuration 188
Monitored property 38
MS SQL Server DataServers
actions allowed on 69
broker properties 70
configuration overview 69

MS SQL Server DataServers *(continued)*
creating instances *76*
deleting instances *77*
editing a configuration *76*
environment variables *76*
server properties *74*
starting
automatically *77*
manually *77*
status *78*
viewing log files *79*

N

NameServer logging level property *86*
NameServers
actions allowed on *82*
configuration overview *82*
configuring local instances *83*
configuring remote instances *89*
creating instances *91*
deleting instances *91*
editing a configuration *90*
environment variables
setting and deleting *90*
features
fault tolerance *92*
load balancing *92*
location transparency *93*
firewalls and UDP *95*
local and remote *83*
starting
automatically *93*
manually *93*
status *94*
viewing log files *96*
Nap interval property *30*
Nap max interval property *30*
Nap time increment property *30*
Nap time steps between nap time property *30*
Neighboring NameServers property *88*
Normal startup arguments property *53*
Number of open SQL cursors property *36*
Number of servers property *41*

O

OE Web Server
Configuring and managing *169*
OE Web Servers
Managing *171*
OpenEdge databases
actions allowed on *20*
assigning a default configuration *39*
background writers *45*
basic properties *22*
OpenEdge databases *(continued)*
configuration overview *20*
configuring replication-enabled *48*
conmgr.properties file *20*
creating a configuration *39*
creating a server group *43*
default configuration
editing *38*
default properties *23*
deleting references *47*
managed *20*
managing instances *21*
process details *46*, *124*
property settings
awareness *21*
basic *22*
default *23*
scripted *20*
server group properties
editing *43*
shared memory systems *44*
starting
automatically *44*
manually *44*
status *45*
stopping *44*
viewing log files *21*
watchdog process *45*
OpenEdge Explorer
comparing OpenEdge Management *15*
introduction *13*
overview *14*
OpenEdge Management
comparing OpenEdge Explorer *15*
introduction *13*
overview *14*
OpenEdge Web Services
configuration overview *170*
default property settings *176*
enabling or disabling for access *180*
managing *180*
managing with the WSA *171*
run-time statistics *181*
undeploying *184*
updating
property settings *183*
Operating mode property
AppServer *101*
MS SQL Server DataServer *70*
Oracle DataServer *59*
WebSpeed Transaction Server *145*
Oracle DataServers
actions allowed on *58*
broker properties *58*
configuration overview *57*
creating instances *65*

Oracle DataServers *(continued)*
deleting instances *65*
editing a configuration *65*
environment variables *64*
server properties *62*
starting
automatically *66*
manually *66*
status *66*
viewing log files *68*
Other arguments property
replication
agent *49*
control agent *52*
server *50*

P

Page writer max buffers property *31*
Page writer queue delay property *31*
Page writer queue minimum property *31*
Page writer scan delay property *31*
Page writer scan property *31*
Password confirm property
AppServer broker *101*
AppServer TLS *116*
WebSpeed broker *145*
Password property
AppServer *101*
MS SQL Server DataServer *71*
Oracle DataServer *59*
WebSpeed Transaction Server *145*
Password to access the key/certificate Confirm property
WebSpeed TLS *160*
Password to access the key/certificate property
AppServer *116*
WebSpeed Transaction Server *160*
Pending connection timeout property *31*
PICA queue nap time property *29*
Pin shared memory segments property *32*
Polling delay maximum property
replication
agent *49*
server *50*
Polling delay minimum property
replication
agent *49*
server *50*
Port number property
AppServer *101*
MS SQL Server DataServer *71*
NameServer *84*
Oracle DataServer *59*
WebSpeed Transaction Server *145*
Port property *52*
Priority weight property
AppServer *105*
MS SQL Server DataServer *74*
Oracle DataServer *62*
WebSpeed Transaction Server *149*
Private key/Digital certificate alias name property
AppServer *116*
WebSpeed Transaction Server *160*
Process details
OpenEdge database *46*, *124*
Process time statistic *47*, *125*
Program Properties *188*
PROPATH property
AppServer *110*
MS SQL Server DataServer *75*
Oracle DataServer *63*
WebSpeed Transaction Server *153*
Properties
AppServer
agent *109*
broker *100*
messaging *116*
TLS *115*
AppServer Internet Adapter *127*
MS SQL Server DataServer
broker *70*
server *74*
OpenEdge database
basic *22*
conmgr.properties file *20*
default *23*
server group *40*
setting awareness *21*
Oracle DataServer
broker *58*
server *62*
WebSpeed Messenger *138*
WebSpeed Transaction Server
agent *153*
broker *144*
messaging *160*
TLS *159*
Proxy host property *175*
Proxy password property *175*
Proxy port property *175*
Proxy username property *175*

R

Register with NameServer property
AppServer *101*
MS SQL Server DataServer *71*
Oracle DataServer *59*
WebSpeed Messenger *139*
WebSpeed Transaction Server *146*

Registration host name property
AppServer 102
MS SQL Server DataServer 72
Oracle DataServer 60
WebSpeed Transaction Server 146
Registration mode property
AppServer 102
MS SQL Server DataServer 72
Oracle DataServer 60
WebSpeed Transaction Server 146
Registration retry property
AppServer 106
MS SQL Server DataServer 74
Oracle DataServer 62
WebSpeed Transaction Server 150
Replication method property 52
Replication-enabled databases
property settings
agent 49
categories 49
control agent 52
rules 48
server 50
transition 53
Reporting interval property 41
Request timeout property
AppServer 106
WebSpeed Transaction Server 150
Resident size statistic 46, 124
Responsibility property 53
REST application
Undeploying 184
Updating a deployed REST application 183
REST applications
TLS support 184
Restart after transition property 53

S

Schema Lock Action property 50
Scripted databases 20
Semaphore sets property 31
Server application mode property 154
Server code page property 29
Server executable file property
AppServer 109
MS SQL Server DataServer 75
Oracle DataServer 63
WebSpeed Transaction Server 153
Server log file threshold size property
AppServer 113
WebSpeed Transaction Server 157
Server log filename property
AppServer 110
MS SQL Server DataServer 75
NameServer 85
Server log filename property *(continued)*
Oracle DataServer 64
WebSpeed Transaction Server 154
Server logging entry types property
AppServer 111
WebSpeed Transaction Server 155
Server logging level property
AppServer agent 111
AppServer messaging 118
WebSpeed agent 155
WebSpeed messaging 162
Server port maximum property 50
Server port minimum property 50
Server startup parameters property
AppServer 110
MS SQL Server DataServer 75
Oracle DataServer 63
WebSpeed Transaction Server 153
Server startup timeout property
AppServer 106
WebSpeed Transaction Server 150
Server Transition property 50
Server Transition Timeout property 50
Server watch dog interval property
AppServer 113
WebSpeed Transaction Server 157
ServerGroup group 20
Service name or port property 41
Session connection ID property 142
Session-Free
AppServers 98
Session-Managed
AppServers 98
Shared memory overflow size property 32
Shared memory systems 44
Shutdown property 114
SonicMQ ServerConnect broker log filename property
AppServer 117
WebSpeed Transaction Server 161
SonicMQ ServerConnect enabled property
AppServer 117
WebSpeed Transaction Server 161
SonicMQ ServerConnect server log filename property
AppServer 117
WebSpeed Transaction Server 161
Source startup arguments property 53
source.repl.properties file 48
Spin lock retries property 32
SQL sorting memory property 36
SQL sorting on disk 36
SQL stack size property 36
SQL statement cache size property 36
SQL temporary data page size property 37
Starting
AppServers
automatically 120

Starting *(continued)*
AppServers *(continued)*
manually *119*
MS SQL Server DataServers
automatically *77*
manually *77*
NameServers
automatically *93*
manually *93*
OpenEdge databases
automatically *44*
manually *44*
Oracle DataServers
automatically *66*
manually *66*
WebSpeed Transaction Servers
automatically *164*
manually *163*
Starting Batch Program *189*
Startup parameters property
AppServer *115*
Startup property *114*
Status
AppServer *121*
MS SQL Server DataServer *78*
NameServer *94*
OpenEdge database *45*
Oracle DataServer *66*
Stopping
OpenEdge databases *44*
Storage object cache size property *32*
Structure file containing After image area definitions property *53*
Supports default service property
AppServer *102*
MS SQL Server DataServer *72*
Oracle DataServer *60*
WebSpeed Transaction Server *146*
Synchronous or Asynchronous property *52*

T

Table range size property *36*
Target startup arguments property *53*
target.repl.properties file *48*
TCP/IP version property
AppServer *107*
NameServer *88*
OpenEdge database *42*
WebSpeed Transaction Server *151*
TLS session cache timeout property
AppServer *116*
WebSpeed Transaction Server *160*
Transition to Agents property *53*
TXE commit lock skip limit property *32*

U

ubroker.properties file *15*
Unified Broker products *82*
Updating, See Changing
URL property *174*
User MUX latches property *33*
User time statistic *46*, *124*
Username property
AppServer *101*
MS SQL Server DataServer *71*
Oracle DataServer *59*
WebSpeed Transaction Server *145*

V

Viewing
AppServer
agent properties *109*
broker properties *100*
environment variables *118*
log files *126*
messaging properties *116*
status *121*
TLS properties *115*
AppServer Internet Adapter
log files *134*
AppServer Internet Adapter properties *127*
deployed Web service
run-time statistics *181*
MS SQL Server DataServer
broker properties *70*
environment variables *76*
log files *79*
server properties *74*
status *78*
NameServer
environment variables *90*
log files *96*
properties *83*
status *94*
OpenEdge database
basic properties *22*
default properties *23*
log files *21*
process details *46*, *124*
server group properties *40*
status *45*
Oracle DataServer
broker properties *58*
environment variables *64*
log files *68*
server properties *62*
status *66*
properties *174*

Viewing *(continued)*
WebSpeed Messenger
log files *142*
properties *138*
WebSpeed Transaction Server
agent properties *153*
broker properties *144*
environment variables *162*
messaging properties *160*
TLS properties *159*
Viewing OE Web Server instance statistics *177*
Viewing the status of an OE Web Server instance *177*
Virtual size statistic *46*, *124*

W

Watchdog process *45*
Watchdog process property *26*
WebSpeed
additional management *166*
WebSpeed Messengers
configuration overview *137*
editing a configuration *142*
properties
advanced features *142*
controlling NameServer *139*
general *139*
logging setting *140*
TLS *141*
viewing log files *142*
WebSpeed transaction server host property *140*
WebSpeed transaction server port property *140*
WebSpeed Transaction Servers
actions allowed on *144*
additional management *166*
WebSpeed Transaction Servers *(continued)*
agent properties
advanced features *158*
general *153*
logging setting *154*
pool range *157*
agents
adding or trimming *164*
broker properties
advanced features *149*
AppService Name List *146*
controlling NameServer *146*
general *145*
owner information *145*
configuration overview *143*
creating instances *163*
deleting instances *165*
editing a configuration *163*
environment variables *162*
messaging properties *160–161*
starting
automatically *164*
manually *163*
TLS properties
advanced features *160*
general *159*
Weighted CPU statistic *46*, *124*
Windows Event Level property *30*
Working directory property
AppServer *101*
MS SQL Server DataServer *70*
NameServer *85*
Oracle DataServer *59*
WebSpeed Messenger *139*
WebSpeed Transaction Server *145*