Sterling Integrator®

Working with Property Files

Version 5.1

Contents

Working with Property Files

Property files contain values that control the operation of the application. By modifying the settings of these properties, you can customize the application to suit your business needs. Property files are located in the *install_dir*/properties (UNIX) directory or the *install_dir*\properties (Windows) directory and are usually named in the following manner: *filename*.properties. Some files have other suffixes including .xml, .xsl, .cfg, and .ini. Substitute the appropriate suffix for properties when needed.

Caution: Since property files directly affect the operation of the application, ensure that you fully understand the impact of property file changes. When changing configuration files, be sure that you have a complete backup of the application system and have fully tested the changes in a test or development environment before moving the changes into production. In addition, some properties should only be changed by Customer Support. For information about working with any property file or property, see the comments in the property file or contact Customer Support.

Property File Types

The following table lists different kinds of property files:

File Type	Description
*.properties	A file that is used during the operation of the application. The initial properties in this file are set by the file's corresponding *.properties.in file.
*.properties*_ext	A file that is used during the operation of the application. It is an extension of the similarly named *.properties file. More application-specific customization can be done in *.properties*_ext files. The initial properties in this file are set by the file's corresponding *.properties*_ext.in file.
*.properties.in	An initialization file that is used during an installation. It sets the initial values of *.properties files.
*.properties*_ext.in	An initialization file that is used during an installation. It sets the initial values of *.properties*_ext files.

File Type	Description
customer_overrides.properties	The file that maintains changes in *.properties and *.properties*_ext files. This file overrides *.properties.in and *.properties*ext.in files when the application is re-installed or when the setupfiles script is run.

Initial Settings for Property Files

In the application, property files are first generated during the installation in the *install_dir*/properties (UNIX/Linux) directory or the *install_dir*\properties (Windows) directory. Their values come from initialization files that are shipped with the application. The initialization files are stored in the same directory as the operational property files.

For example, the property settings in the jdbc.properties.in file are used to create or modify the settings in the jdbc.properties file. During operations, the application references the jdbc.properties file.

With very few exceptions, if you need to override or change the value of a property, you will edit the customer_overrides.properties file to do so. Do not modify or change any properties in files ending with .in because newer versions or patches of the product will overwrite your changes. Do not change a property file that has a corresponding .in file because the setupfiles script will re-create the properties file again, thus causing you to lose your changes. If you need to override or change the value of a property, see the topic *Overriding Properties*.

For example, if you change the contents of the jdbc.properties.in file and then run setupfiles.sh (UNIX/Linux or iSeries) or setupfiles.cmd (Windows), the change will be applied to the jdbc.properties file. If you made this change directly in the jdbc.properties file, the change would be lost during a patch or upgrade installation, or during a system restart that uses the setupfiles command.

After you run the setupfiles command, the old versions of the property files that have been updated are stored in the (UNIX/Linux) *install_dir*/properties/backups directory or the (Windows) *install_dir*\properties\backups directory.

Overriding Properties

Property changes can be required to configure properties, tune performance, or for other reasons. In these situations, use the options described below.

- customer_overrides.properties – Make changes to this override file instead of individual property files to ensure that none of your changes are lost during an upgrade or patch installation. This also lets you see most of the overrides in your configuration at a glance rather than having to check each property file individually. This file is not delivered with the product; you must create it the first time you have a need for it.

  Caution: Do not use the customer_overrides.properties file to override property values for the following files:

  - archivethread.properties
  - security.properties
  - tuning.properties
  - ui.properties

  These files contain specific parameter settings needed for the system to function properly. Do not modify any of these parameters without instruction from Customer Support. See the comments in each file for information on each parameter's function.
- sandbox.cfg – Contains name-value parameters that are merged with each *.in file to create the final properties files.

Using the customer_overrides.properties File

A customer override properties file can be used to override default property settings in the property files. The customer override properties file is not changed during installation of upgrades or patches. To prevent having your customized settings overwritten, use overrides whenever possible rather than editing the property files.

If you have made changes to property files in a previous version, either directly or by editing the associated .in files, your changes may be overwritten when a patch is applied. To prevent this, create a customer_overrides.properties file.

For example, to set the value of the bp_response_timeout property in http.properties so that it is not affected by the bp_response_timeout property in http.properties.in when the setupfiles script is run, add the following line to the customer_overrides.properties file:

```
http.bp_response_timeout=value
```

In this example, http represents the http.properties file, bp_response_timeout represents the bp_response_timeout property, and *value* is the value that you want to prevent the setupfiles script from changing.

You do not need to run the setupfiles script after making a change to the customer_overrides.properties file. However, you do need to stop and restart the application for the changes to take effect.

At this time, no audit trail is provided for changes made to the customer_overrides.properties file.

Before You Begin

For each property that you want to override, you must have the following information:

- PROPERTY_FILE_NAME_PREFIX - The name used in the servers.properties file to reference the actual property file.
- PROPERTY_NAME - The name of the property in the specified property file.
- PROPERTY_VALUE - The value you want to assign to the property.

Tip: Locate a Property File Name Prefix

To find the PROPERTY_FILE_NAME_PREFIX for a property file:

1. Open the properties/servers.properties file and find the entry for the desired property file.
2. The part of the entry before the equal sign (=) is the prefix that you will use in customer_overrides.properties. Make note of it.

   For example, locate the entry for jdbc.properties in servers.properties:

   ```
   jdbcService=install_dir/install/properties/jdbc.properties
   ```

   jdbcService is the prefix for the jdbc.properties property file that you would use in the customer_overrides.properties file.

Create an Entry in the customer_overrides.properties File

To create an entry in the customer_overrides.properties file:

1. Open or create the *install_dir*/properties/customer_overrides.properties (UNIX/Linux) or *install_dir*\properties\customer_overrides.properties (Windows) file.
2. Add the properties that you want to override, using the following format:

   ```
   PROPERTY_FILE_NAME_PREFIX.PROPERTY_NAME=PROPERTY_VALUE
   ```

   PROPERTY_FILE_NAME_PREFIX - Name used in the servers.properties file to reference the actual property file.

   PROPERTY_NAME - Name of the property as specified in the property file.

   PROPERTY_VALUE - Value you want to assign to the property.

For example, assume that you want to change the maximum number of database connections to use in starting up the services controller to 50. To do so, override the maxDatabaseConnections property value in the noapp.properties file by adding the following line to the customer_overrides.properties file:

```
noapp.maxDatabaseConnections=50
```

3. Save and close the customer_overrides.properties file.
4. Stop and restart the application.
5. Test your changes to ensure that the overrides give the desired results. If you have problems, contact Customer Support for assistance.

Using the sandbox.cfg File

The sandbox.cfg file contains name-value parameters that are merged with each *.in file to create the final properties files. Properties in the *.in files that pull their values directly from the sandbox.cfg file are identified with parameters that are contained within the '&' and ';' characters. For example, the jdbc.properties.in file contains the following property:

```
oraclePool.user=&ORA_USER;
```

The &ORA_USER; signifies a parameter. If the sandbox.cfg file contains the entry ORA_USER=oracle, the resulting jdbc.properties file will contain the following property:

```
oraclePool.user=oracle
```

You can edit the information in the sandbox.cfg file at any time to change values that have been created by the installer or to reflect changed setup parameters. For example, if you needed to change a database host name, you would edit that host name in sandbox.cfg and run setupfiles to distribute the new host name into jdbc.properties.

Most of the parameters in the sandbox.cfg file and the *.in files are not used at runtime by the product. Consequently, if you change a parameter in the sandbox.cfg file, you must run the setupfiles script so that the runtime property files are re-created with the updated values.

See the comments in the individual property files for the properties that can be changed and a description of each.

Sample Configuration Properties

This is a sample of the properties in a typical sandbox.cfg file. The file in your installation will vary according to your configuration.

Property	Description
ACTIVEMQ_PORT	Port used by the application for the active WebSphere MQ JMS. It can be sequentially assigned or user-specified. Example: 60847
ADMIN_PORT	

Property	Description
AGENT_JAVA_HOME	JDK directory for the installation of the application. Overrides the Java that the agents use. For example, if, you wanted to use the IBM JDK with WebSphere, you could set this parameter to a new value. Example: *install_dir*/install/jdk
ANONY_PORT	
ant.install.config.version	Example: 0.0
ANT_DIR	Contains the ant binaries used in the Java deployer and other deployment and build scripts. Example: *install_dir*/ant
ANT_VER	Version of the ant released with the application. Example: 1_6_5
APP_DIR	Application server directory. Example: *install_dir*/noapp
APP_SPEC_DIR	Application server-specific directory. Example: *install_dir*/noapp (only current value)
APPBEANS_DIR	Directory that holds jar files for use in a Java 2 Enterprise Edition container. These jar files contain classes required for the EJB Adapter. This allows EJBs to call out to ASI and invoke a business process. Example: *install_dir*/client/ejb
APSERV_INTEGRATE	Installer question indicating whether EJB Adapter will be used to integrate with an application server. Example: No
APSERVER_DIR	Directory for an application server. Example: *install_dir*
ARCHIVE_DB_POOL	Database pool used by the archive process. Example: mysqlArchivePool
AS2_UI	Flag indicating whether this is the AS2 edition of this product. Valid values: • false - The AS2 edition is not installed. • true - The AS2 edition is installed. Default: false
AUDIT_LOAD_DEFAULTS	When set to true, audits are generated when the loadDefaults script is run. Valid values: true, false Example: AUDIT_LOAD_DEFAULTS=false

Property	Description
B2B_FTP_PORT	Port reserved for the default instance of the application FTP Server (base port + 32). Example: 60832
B2B_HTTP_PORT	Port on which the B2B_HTTP_SERVER_SERVICE instance of the B2B HTTP Server Adapter listens to inbound HTTP requests. This port is sequentially assigned. **Note:** You may change this port number. However, you must also change the corresponding web.xml entry in *install_dir*/container/Applications/b2bhttp/WEB-INF/web.xml.in before restarting the system. Example: 60806
BACKUP_OPS_PORT	Application clustering port pointing to the backup operation server process. Port can be sequentially assigned or user-specified. Example: 60828
basedir	(Windows only) Default installation directory. Example: /var/tmp/antinstall12
BIN_DIR	Shell scripts on UNIX and command scripts on Windows. Example: *install_dir*/bin
BOPF_DIR	Directory that contains business object definitions and tools for extending and redeploying them. Example: *install_dir*/busobjs
BPDEFS_DIR	This is exported in the tmp.sh command. Example: *install_dir*
BPEL_JETTY_PORT	Example: 46954
BPMETA_DIR	Location of business process metadata XML descriptors during installation of BPML files. Example: *install_dir*/installed_data
BUILD_NUMBER	Product build number, which indicates the version and patch level of the product. This property is not user configurable. Example: 2000
CDSVR_GIS_PORT1	Assigned Connect:Direct Server Port value for the default out-of-the-box Connect:Direct Server Adapter instance. Example: 60829
CEUSVR_GIS_PORT	Assigned Listen Port value for the default out-of-the-box Connect:Enterprise Unix Server Adapter instance. Example: 60830

Property	Description
CFG_TP	Example: No
CHANGE_DEFAULT_PORTS	Installation question specifying whether the customer needs to configure specific ports or accept the defaults derived from the base port. Example: n
CLA2_PORT	Port reserved for the CLA2 (Command Line 2) Windows Service. It can be sequentially assigned or user-specified. Example: 46952
CLASS_DIR	Location of Java archive (jar) files. Example: *install_dir*/jar
CLIENT_PORT	Port reserved for use by the application. It can be sequentially assigned or user-specified. Example: 60817
CONFIG_GS	Example: No
Continue	Indicates whether to continue or quit the installation. Example: y
CS_PATCH_ROW_THRESHOLD	Indicates whether to circumvent the row update and index creation during a patch installation process. Adding this parameter requires that you manually install the correlation enhancements once you install the patch. Example: 1
CUR_JBOSS_VER	Specifies which version of the Xerces jars is being used. When there were multiple JDKs, they required different versions of these jars. Example: jboss-3.2.1_tomcat-4.1.24
Database	Specifies which database is being used. Format is Database=true. Valid value: Any supported database ID (Oracle, MySQL, MSSQL, DB2, DB2ISERIES, Informix Example: MySQL=TRUE
*Database*_AIX	Binary name for AIX. Example: MYSQL_AIX=mysql-pro-5.0.23-aix5.2-powerpc-64bit
*Database*_CLIENT	Driver for JDBC connectivity. Example: MYSQL_CLIENT=mysql-connector-3.1.13-stable-bin.jar
*Database*_DATA	Database connection information, where DATA is the schema or catalog name of the database. *Database* is a supported database like ORA, MSSQL, etc. Example: MYSQL_DATA=name

Property	Description
*Database*_HOST	Database connection information, where HOST is the hostname or IP of the database server. *Database* is a supported database like ORA, MSSQL, etc. Example: MYSQL_HOST=localhost
*Database*_HPUX	Binary name for HP-UX. Example: MYSQL_HPUX=mysql-pro-5.0.23-hpux11.11-64bit
*Database*_LINUX	Binary name for Linux. Example: MYSQL_LINUX=mysql-pro-5.0.23-linux-i686
*Database*_PASS	Database connection information, where PASS is the user password. *Database* is a supported database like ORA, MSSQL, etc. Example: MYSQL_PASS=password
*Database*_PORT	Database connection information, where PORT is the port used to connect to the database. *Database* is a supported database like ORA, MSSQL, etc. Example: MYSQL_PORT=46903
*Database*_SOLARIS	Binary name for Solaris. Example: MYSQL_SOLARIS=mysql-pro-5.0.23-solaris8-sparc-64bit
*Database*_USER	Database connection information, where USER is the user name for connecting to the database. *Database* is a supported database like ORA, MSSQL, etc. Example: MYSQL_USER=si
*Database*_WIN	Binary name for Windows. Example: MYSQL_WIN=mysql-pro-5.0.23-win32
DAV_PORT	Port used by the WEB_DEV server to retrieve components such as Eclipse plug-ins, jar files, etc. Used by Eclipse and the Reporting Services event listeners, but there might be other users of this port. Example: 60846
DB_CREATE_SCHEMA	Example: yes
DB_DATA	Used at installation only. Database name to connect to. **Note:** To change this database property after installation, use the database-specific properties. Example: name
DB_DRIVERS	Used at installation only. Full path to the location of the JDBC drivers specified during the installation. **Note:** To change this database property after installation, use the database-specific properties.

Property	Description
	Example: *install_dir*/mysql/driver/mysql-connector-3.0.8-stable-bin.jar
DB_DRIVERS_VERSION	Version of the database drivers. Enter a string representing the version as you recognize it. Functionally used only to build a directory structure for the database drivers for install3rdParty functionality. Example: 2_0_14
DB_HOST	Used at installation only. Database host to connect to. **Note:** To change this database property after installation, use the database-specific properties. Example: localhost
DBINIT_DIR	
DB_JAR_DIR	Used at installation only. The location of the database JDBC drivers. This is referenced in the dynamicclasspath.cfg files to put the drivers in the classpath. **Note:** To change this database property after installation, use the database-specific properties. Example: *install_dir*/dbjar
DB_PASS	Database password to connect with. DB_PASS=userpassword
DB_POOL	Database pool used in various parts of the system, named as *database*Pool where *database* is mysql, oracle, etc. **Note:** To change this database property after installation, use the database-specific properties. Example: mysqlPool
DB_PORT	Used at installation only. Database listener port. **Note:** To change this database property after installation, use the database-specific properties. Example: 3306
DB_SCHEMA_OWNER	Default schema or schema-owner for the provided login ID. DB_SCHEMA_OWNER=username
DB_USER	Database login ID to connect with. **Note:** To change this database property after installation, use the database-specific properties. Example: DB_USER=*username*

Property	Description
DB_VENDOR	Required. The database vendor used for the application database (MySQL, MSSQL, Oracle, etc.). **Note:** To change this database property after installation, use the database-specific properties. Example: MySQL
DBDIST_DIR	This is exported in the tmp.sh command. Example: *install_dir*
DEBUG_OPS_PORT	Port used to connect a remote debugger to the ops process. Example: 60805
DEBUG_PORT	Port used to connect to the noapp JVM with a remote debugger.
DEPLOYED_APP_DIR	Directory used to deploy Web applicatons. Example: *install_dir*/noapp/deploy
DIST_DIR	This is exported in the tmp.sh command. Example: *install_dir*
DOC_DIR	This is exported in the tmp.sh command. Example: *install_dir*
EBXML_HTTP_SERVER_PORT	Example: 46955
EDITEST_DIR	Example: *install_dir*/install
EVENT_PORT	Port that can be used for non-JVM producers and consumers of events. This is used in an event property file. Example: 21258
EXT_HOST_ADDR	External IP address of the host, used by Web Start applications such as the GBM to be made available for use on an external IP address. Example: *ip_address*
FARM_DIR	Not used. References a directory used in JBoss clustering.
FEDERATION_HTTP_SERVER_PORT	Listening port for the out-of-the-box HTTP Server Adapter instance in this application that hosts the Federation application. In a normal installation, this port will be offset +37 from the main port number of the application. Example: 60837
FIPS_MODE	Indicates whether FIPS (Federal Information Processing Standards) mode is enabled. Default: no

Property	Description
FTP_ACCT_PORT	Port reserved for regression test of the application FTP Client for testing the remoteAccount parameter (base port + 45). Example: 60845
GIS_JAR	Location of the application installation jar file. Example: *install_dir*/product/IFC-3000.jar
GS_LIFE	Example: No
HOME_DIR	Not used. Used in various scripts for compatibility with the sandbox. Derived from the INSTALL_DIR property. Example: *install_dir*
HOST_ADDR	IP address of the machine containing the application installation. Example: *ip_address*
HOST_NAME	Name of the machine containing the application installation. Example: *serverName*
HSQL_PORT	Not used.
HTTP_SERVER_PORT	The listen port for the primary out-of-the-box HTTP Server Adapter in the application named "HttpServerAdapter." In a normal installation, this port is offset +33 from the main port number of the application. Among others, this adapter instance is configured with the /dashboard url which points to the main dashboard WAR file. Example: 60833
HYPER_PORT	Not used.
Icons	Example: false
INSTALL_DIR	Path to the application installation directory. This is the root of the directory structure for the application on the file system. This is a user-specified value. This is used extensively in .in files to munge the correct root installation directory. Also used in scripts in the main installation as well as in regressions.
INSTALL_IP	IP address for installation. Example: localhost
JAR_DIR	Directory used by install and install3rdParty to store 3rd party software jar files (referenced by the dynamic class loader and tmp.sh for the java classpath). Example: *install_dir*/jar

Property	Description
JAR_JAVA_HOME	Points to the location of the Java SDK so that the jar utility can be found. This entry is derived from the user-specified INSTALL_DIR property. Example: *install_dir*/jdk
JASPER_VER	Example: 1_0_0
JASPER_VER_DOT	Version of the Jasper Reports. Example: 1.0.0
JAVA_HOME	Points to the location of the Java SDK used for the application installation. This property is derived from the user-specified INSTALL_DIR property. Example: *install_dir*/jdk
JAVADOC_PRODUCT_LABEL	Example: Platform Javadocs
JCE_DIST_FILE	Path and file name of the JCE (Java Cryptography Extension) file used for the application installation. This is the location of the "unlimited strength" JCE inserted into the JDK during installation. The value of this property is user-specified. Example: *home_dir*/jce/1_4_2/jce_policy-1_4_2.zip
JDBC_DRIVER	Path and file name of the JDBC database driver. Example: *install_dir*/mysql/driver/mysql-connector-3.0.8-stable-bin.jar
JDBC_VENDOR JDBC	Driver vendor. Used when multiple vendors are available. Default=Microsoft
JDBC_VER	Not used. Version of the JDBC driver.
JMX_AGENT_PORT	Example: 46949
JMX_HTML_CONSOLE_PORT	Example: 46951
JMX_RMI_PORT	Example: 46950
JNDI_PORT	Used for Java Naming and Directory Interface lookups. Used by workflow and ops processes and in code to configure the JNP server. Example: 60813
JVM_LOC	Source of JDK files. Example: *install_dir*
JVM15	Differentiates between JVM versions used to build this application. This property is set to either true or not present if a JVM other than 1.5 was used to build the installation image. Default: true
LIC_PROD_VERSION	Product version (not build version). Example: 2.0

Property	Description
LICENSE_FILE_PATH	Path and file name of the license file for this installation. The value is user-specified. Example: *install_dir*/Full_License_Dev.xml
LIST_PORT	Base port for the application installation. The starting port in a range of ports reserved for use by the application. Same as PORT1. Example: 60800
LOAD_FACTORY_SETUP	Indicates whether to load factory setup defaults during installation or manually after installation. Valid values: true, false Example: LOAD_FACTORY_SETUP=true
LOCAL_JNDI_PORT	Port reserved for use by the application. It can be sequentially assigned or user-specified. Example: 60814
LOCALHOST	Example: localhost
LOG_DIR	Path to the subdirectory where log files are stored. Use to override the logging directory. For example, to deploy the ear on another server, set the parameter to a new value, run setupfiles, then build the ear. The application would then log to the directory you specified. Example: *install_dir*/logs
LOG4J_VER_DOT	Example: 1.2.11
MAPTEST_HTTP_SERVER_PORT	Port on which the Map Test HTTP Server Adapter listens to process inbound requests from the Map Editor's Map Test utility. This port can be sequentially assigned or user specified. Example: 60838
MBI_HTTP_SERVER_PORT	The listen port for the out-of-the-box HTTP Server Adapter instance in application that hosts the MBI (Mailbox Browser Interface) and the myAFT (Advanced File Transfer) external portals. In a normal installation, this port will be offset +34 from the main port number of the application. Example: 60834
MULTICAST_NODE_PORT*n*	Base port for multicast communication between nodes. Example: 60839
NEO_HTTP_SERVER_PORT	Port reserved for use by the application. It can be sequentially assigned or user-specified. Example: 60836

Property	Description
NO_DBVERIFY	When set to true during installation and installservice, dbverify will not be run and the DDL to make the database like the XML entity repository will not be generated. Valid values: true, false Example: NO_DBVERIFY=false
NOAPP	A legacy value that differentiated between the different types of application servers that the application used to support. This property is always present and has a value of true. Many legacy files still require this variable to be present. Default: true
NOAPP_DIR	Points to the noapp directory which contains the web applIcation deployment directory and various scripts. Example: *install_dir*/noapp
NOAPP_HOME	Path to the noapp directory in an installation. This directory houses the custom application server (ASI or noapp). This is referenced in many other files to gain path information to scripts and other noapp directories for classpaths. Example:*install_dir*/noapp
NODE_NAME	Used in clustering to name this node. Default: node1
OPS_PORT	Operations server port. Example: 60827
ORACLE_USE_BLOB	Indicates whether the BLOB (binary large object) data type or the Long Raw data type is being used in an Oracle database. Used in several property files to indicate which IVarData implementation class to use for various connection pools.
PARTITION_NAME	Example: *serverName*_60800_Partition
PORT1	Base port for the application installation. Starting port in a range of ports reserved for use by the application. This is a user-specified value. The remaining ports can be sequentially assigned or can be user-specified. Example: 21200
PORT2	Used for SSL. Example: 60801
PRE_INSTALL_DIR	Directory where PreInstallSI.log file is stored during the application installation. This is a temporary file that tracks the progress of an installation. This file can be used for troubleshooting. Example: *install_dir*

Property	Description
PRE_INSTALL_LOG_DIR	Directory path and file name of the temporary file that tracks the progress of an installation. This file can be used for troubleshooting. Example: install_dir/PreInstallSI.log
PRODUCT_LABEL	Platform release used with the application. Example: Platform_2.0
PRODUCT_NAME	Determines the subdirectory where certain jars are shipped. For example, the file path jars/platform/2.0/Security.jar would use PRODUCT_NAME=platform.
PROG_DIR	Path to the *install_dir*/container/Applications directory that holds the various .war files deployed on the ASI/noapp server. Example: *install_dir*/container/Applications
PROP_DIR	Path to the properties subdirectory for the application installation. Example: *install_dir*/properties
properties.18-targets	Example: Disk Space-sun,OS Check,JDK Version Check,Core Components on-sun,Save install files-sun,
properties.3-targets	Example: Install Directory,
REINIT_DB	Indicates whether database updates are repeated for each node of a cluster installation. Valid values: true (database updates are repeated), false Example: REINIT_DB=true
RES_PROP_DIR	Not used. Directory used for creating a jar file of resource and properties files to be deployed to the JBoss classpath.
RMI_PORT	Port reserved for use by the application. It can be sequentially assigned or user-specified. Example: 60812
RN_HTTP_SERVER_PORT	Port on which the RosettaNet HTTP Server Adapter listens to process inbound requests. This port can be sequentially assigned or user-specified. Example: 60835
SFTP_SERVER_PORT	Assigned SFTP Server Listen Port value for the default out-of-the-box SFTP Server Adapter instance. Example: 60839
SI_ADMIN_MAIL_ADDR	E-mail address where administrative alerts are to be sent. This is a user-specified value. Example: alert_test@stercomm.com

Property	Description
SI_ADMIN_SMTP_HOST	E-mail server where administrative alerts are to be e-mailed. This is a user-specified value. Example: mail.stercomm.com
SI_LICENSE_AVAILABLE	No longer used. Formerly used by installation to determine whether to load the license set. License is now required during installation.
SN_HTTP_SERVER_PORT	Port on which the SWIFTNet HTTP Server Adapter listens to process inbound SWIFTNet requests. This port can be sequentially assigned or user specified. Example: 60853
SNMP_GENTEST_PORT	Port used in regression testing for SNMP. It is used as a Trap Receiver port. Example: 60822
SNMP_PORT1	Port that is used for regression test for SNMP Trap Receiver (for basic loop). Example: 60823
SNMP_PORT2	Port used for regression test for SNMP Trap Receiver (for Excluded Community). Example: 60824
SNMP_PORT3	Port used for regression test for SNMP Trap Receiver (for Unspecified Community). Example: 60825
SOA_PORT	Port on which the SOA HTTP Server Adapter, a pre-configured instance of the HTTP Server Adapter, listens to inbound HTTP requests to bootstrap business processes for the Web service feature. This port is sequentially assigned. Example: 60840
SOA_SSL_PORT	Port on which the HTTP Server Adapter listens for an incoming SOAP/HTTPS request. This port is sequentially assigned. Example: 60841
SOAP_PORT	Port on which the BaseHttpService instance of the B2B HTTP Server Adapter listens to inbound HTTP requests. This port is sequentially assigned. **Note:** You may change this port number. However, you also must change the corresponding web.xml entry in *install_dir*/container/Applications/webservices/WEB-INF/web.xml.in before restarting the system. Example: 60810

Property	Description
SSL_PORT	Not used.
SSLPLUS_VER	Example: 4_3_5_12
SUPPORT_MULTIBYTE	Example: N
SVC_DIR	Formerly, a directory used by the user interface to load service definitions. Service definitions are now loaded from the database and files are stored in this location. Example: *install_dir*/properties/services
SYSGENWARS_DIR	Derived from the INSTALL_DIR property, which is the user-specified root of the directory structure for the application on the file system. Example: *install_dir*/sysgenwars
SYSTMP_DIR	Derived from the INSTALL_DIR property, which is the user-specified root of the directory structure for the application on the file system. Example: *install_dir*/tmp
TOMCAT_AJP_REDIRECT_PORT	Not used.
TOMCAT_AJP12_PORT	Not used.
TOMCAT_AJP13_PORT	Not used.
TRANSLATOR_VER	Subdirectory that contains the translator.jar file. For example, install_dir/jar/translator/5.0/translator.jar uses TRANSLATOR_VER=5.0.
TRUSTPOINT_VER	Version number of the trustpoint JAR files used in the installation. Used to specify which version of the Trustpoint jars is being used. When there were multiple JDKs, different versions of these jars were required. Example: 3_1_0_7_sci1
UI_DB_POOL	Database pool used by the user interface. Example: mysqlUIPool
UI_JNDI_PORT	Not used.
UI_PORT	Not used.
UI_SSL_PORT	Not used.
UPGRADE	Example: false
UPGRADE_MAJOR_VERSION	Used to establish the minimum product version that can be eligible for in-place upgrade. Default value: 4
UPGRADE_MINOR_VERSION	Used to establish the minimum product version that can be eligible for in-place upgrade. Default value: 0
USE_NEW_INSTALL	Legacy value that differentiates the old installation process from the new installation process. This property is always present and has a value of

Property	Description
	true, indicating the new installation process. However, many legacy scripts still require this variable to be present. Default: true
VENDORS_DIR	Directory used to hold resource jar files. This directory is the same as JAR_DIR and is a legacy name from the sandbox. Example: *install_dir*/jar
WEBX_PORT	Port on which the WEB_EXTENSIONS_HTTP_SERVER_ADAPTER instance of the B2B HTTP Server Adapter listens to inbound HTTP requests. This port is sequentially assigned. **Note:** You may change this port number. However, you also must change the corresponding web.xml entry in *install_dir*/container/Applications/webx/WEB-INF/web.xml.in before restarting the system. Example: 60808
XALAN_VER	Version of the Xalan jars being used. When there are multiple JDKs, different versions of these jars are required. Example: 2_5_2
XERCES_VER	Version of the Xerces jars being used. When there are multiple JDKs, different versions of these jars are required. Example: 2_6_0
YANTRA_DB_DATA	Example: name
YANTRA_DB_HOST	Example: localhost
YANTRA_DB_PASS	Used with the Yantra adapter. Add the following line after installing the Yantra .jar files:YANTRA_DB_PASS=*userpassword*
YANTRA_DB_PORT	Example: 3306
YANTRA_DB_USER	Used with the Yantra adapter. Add the following line after installing the Yantra .jar files: YANTRA_DB_USER=*username*

Modifying Properties in a .properties.in File

Note: Whenever possible, changing properties by changing the customer_overrides.properties file instead of the *.properties.in files is highly recommended.

Although overriding property file settings using the customer_overrides.properties file is usually the best option for setting and keeping a property value, you can also edit property files. Most *.properties files have associated *.properties.in files. If possible, edit properties in the *.properties.in file rather than the *.properties file directly, since the *.properties.in file sets the value of the *.properties file when the setupfiles script is run.

If a *.properties file does not have an associated *.properties.in file, you will need to edit the *.properties file directly.

Change Settings in a .properties.in File

To change settings in a *.properties file by editing its associated *.properties.in file, perform the following steps:

Note: For cluster installations, perform this full procedure for each node.

Note: Leading or trailing whitespace in property files will be respected by the application. This may cause a problem if the system is not expecting white space. When editing property files, be careful to trim leading and trailing whitespace before saving each file.

1. From the *install_dir*/install/properties (UNIX/Linux) or *install_dir*\install\properties (Windows) directory, edit the necessary properties in the *PropertyFileName*.properties.in file.
2. Stop the application.
3. Run the setupfiles script using one of the following steps:
   - UNIX or Linux – From the *install_dir*/install/bin directory, run the setupfiles.sh command.
   - Windows – From the *install_dir*\install\bin directory, run the setupfiles.cmd command.
4. Start the application.

Change Properties in a *.properties File

Note: You should only edit a *.properties file directly if it does not support the use of the customer_overrides.properties file and does not have an associated *.properties.in file.

For cluster installations, perform this full procedure for each node.

Note: Leading or trailing whitespace in property files will be respected by the application. This may cause a problem if the system is not expecting whitespace. When editing property files, be careful to trim leading and trailing whitespace before saving each file.

To change settings in a *.properties file directly, perform the following steps:

1. Stop the application.
2. From the *install_dir*/install/properties (UNIX/Linux) or *install_dir*\install\properties (Windows) directory, edit the necessary properties in the *PropertyFileName*.properties file.
3. Start the application.

Tagged Property Files

Property files include commented tags that document information about the files and their properties.

Property tags are differentiated into the following categories. Each property file must contain one or more of the following types of tags:

- File tags
- Property tags
- Miscellaneous tags

Property File Tags

The following file tags appear at the beginning of a property file:

Tag	Description
PROPERTY_FILE_NAME	Name of the property file
PROPERTY_FILE_DESCRIPTION	Description of the property file.

Property Tags

The following property tags are used once per property in a property file:

Tag	Description
PROPERTY_START, PROPERTY_END	Beginning and End tags of property-specific tags for each property. Helps identify individual properties for inline documentation.
PROPERTY_NAME	Name of the property.

Tag	Description
PROPERTY_GROUP	(Optional) Group that includes property, when you have related properties that can be grouped.
PROPERTY_TYPE	Type of property (String, boolean, integer).
PROPERTY_DESCRIPTION	Description of the property.

Miscellaneous Tags

Tag	Description
PROPERTY_VALUE_ALL	Used to group all property values at the end of a file. For example, related property values that use conditional statements to determine their value may be grouped.

Sample Tagged Property File

The following example shows part of the tagging for the archivethread.properties.in file:

```
## PROPERTY_FILE_NAME
## archivethread.properties.in (for initialization)
## archivethread.properties (for operations)
## PROPERTY_FILE_DESCRIPTION
## The archivethread.properties file is used to control Purge service
functionality.
## The default settings for the properties in the archivethread.properties file
should
## only be modified if you are having problems running the Purge service.
##
## The properties in the archivethread.properties file cannot be overridden.
## Any property changes necessary should be made in the archivethread.properties.in
 file.
##
## CAUTION:
## The Purge service is a critical part of the application and incorrectly-set
parameters
## in the archivethread.properties file can cause problems. Contact Sterling
Commerce
## Customer Support for assistance before modifying any property settings in this
## file.
##
## PROPERTY_START
## PROPERTY_NAME: DETAILS_REPORT_FLAG
## PROPERTY_TYPE: Integer
```

```
## PROPERTY_DESCRIPTION
## If set to 1, provides additional details in the report logs.
DETAILS_REPORT_FLAG=0
## PROPERTY_END
## PROPERTY_START
## PROPERTY_NAME: GENERATE_PURGE_DOCDISK_LIST
## PROPERTY_TYPE: Boolean
## PROPERTY_DESCRIPTION
## Specifies whether to generate a file that lists documents stored
## on disk that are eligible to be removed from the file system.
## Valid values:
## true : Generate a file (default)
## false : Do not generate a file.
GENERATE_PURGE_DOCDISK_LIST=true
## PROPERTY_END
.
.
.
```

Properties for LDAP User Authentication

This section assumes you understand how LDAP servers work. Sterling Commerce also recommends that you read the following documents on LDAP technology:

- W. Yeong, T. Howes, and S. Kille, RFC 1777 - *Lightweight Directory Access Protocol*. March 1995. Available at *http://rfc.sunsite.dk/rfc/rfc1777.html*.
- Mark Wilcox, *Implementing LDAP*. Wrox Press, 1999.

By default, all authentication is performed against the application database. When a user enters a login ID and password, it is validated against the login ID and password that is stored in the database. This requires the administrator of the application system to set up login IDs and passwords for each user.

You may choose to use an LDAP server for authentication. When using LDAP, the users, user groups, and access control must be set up in the application system.

The application also supports password expiration through LDAP. Your custom code for user authentication is interfaced with the application authentication mechanism. If your custom code contains “ExpireInDays” with a numeric value of *X*, then a message to reset the password appears in the application home page. If the map contains “ChangePasswordLink," then the message contains a link to the location specified. Clicking on the link opens a new window with the given “ChangePasswordLink”.

Since the various implementations of LDAP handle password expiration differently, a sample YFSLdapAuthenticator.java file is modified to provide an example of one particular implementation. This is located in the (UNIX/Linux) *install_dir*/install/xapidocs/code_examples/java directory or the (Windows) *install_dir*\install\xapidocs\code_examples\java directory.

Set Properties for LDAP-based User Authentication

To set properties for LDAP-based authentication:

Caution: Change only the properties referred to in this section. Changes to any other properties for the purpose of configuring LDAP security are not supported.

1. Install the LDAP server. See the installation instructions from your LDAP server vendor for details.
2. If a JAAS-compliant provider is used, create a JAAS configuration file with the following lines:

```
LDAP
{
```

```
// refer to the JAAS compliant service provider for the login
    module details.
  <Class Name of the Login Module as specified by the Security
   provider> required
debug=true;
};
```

3. In your customer_overrides.properties file, specify the LDAP properties described in the following table. For more information about the customer_overrides.properties file, see the topic *Overriding Properties Using the customer_overrides.properties File*.

Property	Description
In the customer_overrides.properties file, specify:	
yfs.yfs.security.authenticator	The class that will be invoked for user authentication. Develop a new class that implements the com.yantra.yfs.japi.util.YFSAuthenticator interface and set the new classname as value for this property. If using JASS, set this property value to `com.yantra.interop.services.security`. The application provides a sample com.yantra.yfs.util.YFSLdapAuthenticator class that you can use for reference.
yfs.yfs.security.ldap.factory	If the default implementation is used, this property specifies the LDAP context factory classname as specified in your LDAP server configuration. Set this property value to com.sun.jndi.ldap.LdapCtxFactory.
yfs.yfs.security.ldap.url	If the default implementation is used, this property specifies the URL used to access your LDAP Server. For example, yfs.security.ldap.url=ldap://MyServer:800.
yfs.yfs.security.ldap.o	If the default implementation is used, this property specifies the application organization in your LDAP Server configuration.
yfs.yfs.security.ldap.ou	If the default implementation is used, this property specifies the application organizational unit in your LDAP Server configuration.
yfs.yfs.jaas.loginmodule	If using JAAS, set this property value to LDAP.
WebLogic startWLS startup file	
-Djava.security.auth.login.config	If you are using JAAS and WebLogic, specify the full path to your JAAS configuration file.

Securing the Interop Servlet

The tasks to customize authentication and authorization for the HTTP API servlet include additions to the customer_overrides.properties file. This file allows you to control authentication from within the InteropHttpServlet, which supports both container and token-based authentication.

To configure container and/or token-based authentication, set the following properties in the customer_overrides.properties file:

- yfs.interopservlet.auth.container.enabled=true/false
- yfs.interopservlet.auth.token.enabled=true/false
- yfs.interopservlet.auth.userPassword.enabled=true/false

Caution: Change only the properties referred to in this section. Changes to any other properties for the purpose of securing the interop servlet are not supported.

If every enabled option fails, then the user is not authenticated. The default value is true for the interopservlet.auth.token.enabled property and the interopservlet.auth.userPassword.enabled property.

Note: As a special case, if no authentication mechanisms are enabled, then full access will be granted to the servlet. This should not be done on a production server.

Access to the HTTP API can also be secured via modifications to the deployment descriptor. The deployment descriptor's web.xml is defined by the servlet specification from Sun Microsystems. This deployment descriptor can be used to deploy a Web application on any J2EE-compliant application server. The deployment descriptor for the application is stored in the following directory:

- UNIX/Linux

  ```
  install_dir/repository/eardata/platform/
  descriptors/application_server_type/WAR/WEB-INF
  ```

  Valid values for *application_server_type* are JBoss, WebLogic, or WebSphere.
- Windows

  ```
  install_dir\repository\eardata\platform\
  descriptors\application_server_type\WAR\WEB-INF
  ```

  Valid values for *application_server_type* are JBoss, WebLogic, or WebSphere.

By using the security-constraint element with the web-resource-collection element, you can set up authorization to protect this page from unauthorized access. For more information about the web.xml deployment descriptor, see the documentation for your application server.

Enable Different Properties for Individual Processes

It is possible to specify different properties for each process you are running. To do this, you must have a different servers.properties file and customer_overrides.properties file for each process that you are running. The servers.properties file has information that you use in the customer overrides file.

For example, some of the processes in servers.properties would look like:

noapp=*property_dir*/noapp.properties

shell=*property_dir*/noapp.properties

weblogic=*property_dir*/noapp.properties

websphere=*property_dir*/noapp.properties

jboss=*property_dir*/noapp.properties

jdbcService=*property_dir*/jdbc.properties

The names on the left are abbreviations for process names that are used in the customer_overrides.properties file. The values on the right indicate the files that contain the properties.

To specify different properties for each process:

1. In the start scripts for the process, set the following:
   `-DvendorFile=`*`your_custom_servers.properties`*
2. In your customer servers.properties file, change the entry for customer_overrides.properties to point to your new customer_overrides.properties file.

Properties to Prevent Cross-Site Script Vulnerabilities

In some cases, data to and from the application can contain HTML characters that impact the display and the original intent of the input. In addition, data can be input that contains malicious HTML, such as commands embedded within <SCRIPT>, <OBJECT>, <APPLET>, and <EMBED> tags.

Caution: Change only the properties referred to in this section. Changes to any other properties for the purpose of preventing cross-site vulnerabilities are not supported.

The yfs.htmlencoding.triggers property in the yfs.properties.in file specifies the following characters that could signify potentially unsafe HTML content:

- Greater than symbol (>)
- Less than symbol (<)
- Right parenthesis ( ))
- Right bracket ( ])

If needed, you can add any other characters necessary for your specific implementation to the customer_overrides.properties file.

If information being written to the browser contains any of these characters, the output is safely encoded to prevent exploitation of cross-site scripting vulnerabilities in the application.

For more detailed information about malicious scripts, see the following articles:

- CERT Advisory, Malicious HTML Tags Embedded in Client Web Requests. Available from *http://www.cert.org/advisories/CA-2000-02.html*.
- CERT Advisory, Frequently Asked Questions About Malicious Web Scripts Redirected by Web Sites. Available from *http://www.cert.org/tech_tips/malicious_code_FAQ.html*.

Copyright