ORACLE
INSURANCE

Oracle® Documaker

Docupresentment Features and Enhancements

version 2.2

Part number: E14902-01

May 2009

Oracle Insurance announces a new release of Docupresentment. This document introduces Docupresentment version 2.2 and describes its features and enhancements. In addition, this document includes information development and support policies.

To receive the full benefits of the new product features included in this and earlier releases, Oracle University offers a comprehensive range of training classes. For a list of courses, including fees and availability, please call 404.439.5500.

NOTE: For installation instructions, see the Internet Document Server Installation Guide.

Contents

Monotype Font(s) Software License Agreement

1 "Software" shall mean the digitally encoded, machine readable, scalable outline Intellifont/ data as encoded in a special format as well as the Intellifont Software.

2 You agree to accept a non-exclusive license to use the Software to reproduce and display weights, styles and versions of letters, numerals, characters and symbols ("Typefaces") solely for your own customary business purposes which shall not include use or access by entities other than you. Under the terms of this License Agreement, you have the right to use the Monotype Imaging, Inc., Typefaces only in conjunction with the Docucorp International, Inc., a subsidiary of Skywire Software, L.L.C., Documentation Software licensed by you. Monotype (formerly Agfa) retains all rights, title and interest to the Software and Typefaces and no rights are granted to you other than a License to use the Software on the terms expressly set forth in this Agreement.

3 To protect proprietary rights of Monotype, you agree to maintain the Software and other proprietary information concerning the Typefaces in strict confidence and to establish reasonable procedures regulating access to and use of the Software and Typefaces.

4 You agree not to duplicate or copy the Software or Typefaces, except that you may make one backup copy. You agree that any such copy shall contain the same proprietary notices as those appearing on the original.

5 This License shall continue until the last use of the Software and Typefaces, unless sooner terminated. This License may be terminated by Monotype if you fail to comply with the terms of this License and such failure is not remedied within thirty (30) days after notice from Monotype. When this License expires or is terminated, you shall either return to Monotype or destroy all copies of the Software and Typefaces and documentation as requested.

6 You agree that you will not modify, alter, disassemble, decrypt, reverse engineer or decompile the Software.

7 Monotype warrants that for ninety (90) days after delivery, the Software will perform in accordance with Monotype-published specifications and diskette will be free from defects in material and workmanship. Monotype does not warrant that the Software is free from all bugs, errors and omissions.

8 THE PARTIES AGREE THAT ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE AND MERCHANTABILTY, ARE EXCLUDED.

9 Your exclusive remedy and the sole liability of Monotype in connection with the Software and Typeface is repair of replacement of defective parts, upon their return to Monotype.

10 IN NO EVENT WILL MONOTYPE IMAGING, INC., BE LIABLE FOR LOST PROFITS, LOST DATA, OR ANY OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES CAUSED BY THE ABUSE OR MISAPPLICATION OF THE SOFTWARE AND TYPEFACES.

11 Massachusetts U.S.A. Law governs the Agreement.

12 You shall not sublicense, sell, lease, or otherwise transfer the Software and/or Typefaces without the prior written consent of Monotype.

13 Use, duplication or disclosure by the Government is subject to restrictions as set forth in the Rights in Technical Data and Computer Software clause at FAR 252-227-7013, subdivision (b)(3)(ii) or subparagraph (c)(1)(ii), as appropriate. Further use, duplication or disclosure is subject to restrictions applicable to restricted rights software as set forth in FAR 52.227-19 (c) (2).

14 YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT, AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. NEITHER PARTY SHALL BE BOUND BY ANY STATEMENT OR REPRESENTATION NOT CONTAINED IN THIS AGREEMENT. NO CHANGE IN THIS AGREEMENT IS EFFECTIVE UNLESS WRITTEN AND SIGNED BY PROPERLY AUTHORIZED REPRESENTATIVES OF EACH PARTY. BY OPENING THIS DISKETTE PACKAGE, AND YOU AGREE TO ACCEPT THE TERMS AND CONDITIONS OF THIS AGREEMENT.

Chapter 1

Features and Enhancements

Oracle Insurance proudly announces Docupresentment version 2.2.

This document provides detailed information on the specific features and enhancements to Docupresentment. For a list of the new features and enhancements, see System Enhancements on page 2.

NOTE: For information on installing Docupresentment version 2.2, please refer to the Docupresentment Installation Guide.

System Enhancements

The following pages describe the features which have been included in version 2.2.

As you review the descriptions of the new features, keep in mind that XML standards, as defined by the W3C, require you to substitute text characters that are not in XML tags (for example, between <entry> and </entry> tags) as *escape sequences*. The characters that require substitution are listed in the following table. If you cut and paste an XML example from this or other Docupresentment documentation into an XML configuration file, you will have to manually make these substitutions.

For this character	Use this escape sequence
< (less than)	<
> (greater than)	>
& (ampersand)	&
' (apostrophe)	'
" (quotation mark)	"

Here is a list of the new features:

Feature	License	For more information, see
1749	IDS	Searching a Directory Information Tree on page 3
1782	IDS	Using the New LDAP API on page 12
2198	IDS	Monitoring and Managing IDS Instances on page 12
2247	IDS	Passing Additional JVM Options to DSILIB on page 20
2248	IDS	Generating Debug Output for Client Requests on page 21
2257	IDS	Using Default Time-outs for DSILIB-based Client Applications on page 22
2259	IDS	Running Timed Requests on page 23
2269	IDS	Using the New Java Test Utility on page 24
2278	IDS	Using Client Connection Definition Tables on page 28
2306	IDS	WebSphere MQ Security Exit Support on page 29

NOTE: Docupresentment version 2.2 includes Enterprise Web Processing Services (EWPS). EWPS makes it easier to integrate applications, providing a set of web services for accessing the Documaker forms library and initiating real-time publishing operations. This helps you deliver the information requested by your clients, prospects, employees, and business partners.

For more information on EWPS, see Introduction to Enterprise Web Processing Services.

1749
IDS

SEARCHING A DIRECTORY INFORMATION TREE

IDS now supports searching a Directory Information Tree (DIT) in an LDAP server. The system offers a C rule (DPRSearchLDAP) and a Java rule (search) for conducting LDAP queries to determine a user ID group or role membership. The rules will look for all configuration options in rule arguments, a properties file, INI options, and input attachment variables, in that order. Option values found in more than one source override the previous value.

Input Options

Option	Description
RUNMSG	(Optional) An integer value between 1 and 4 indicating in which message the rule should run: INIT(1), TERM(2), RUNF(3), RUNR(4). This option is only searched as a rule argument or input message variable. The default is 3.
LDAP.PROPERTIES	(Optional) The name of a properties file that should be used to look up the options for the rule. The default file name is *ldap.properties*, which is looked up in the current directory for IDS. This option is only searched as a rule argument or input message variable.
LDAP.HOST	(Optional) The host name or IP address of the LDAP server. The default is localhost.
LDAP.PORT	(Optional) The port in which the LDAP server is listening on. The default is 389 when SSL is not used, 636 otherwise (see LDAP.USE.SSL option).
LDAP.URL	(Optional) The URL the LDAP server is listening on. If a value is specified for this property, it overrides the values specified for LDAP.HOST and LDAP.PORT.
LDAP.UID	(Optional) The user ID for logging into the LDAP server. If this value is provided and LDAP.USER is not provided, the user ID is derived from this value and the value provided for LDAP.DOMAIN option, such as *administrator@pd.com*.
LDAP.USER	(Optional) An explicit value to use for the user ID for the purpose of login into the LDAP server. Define this option to override the behavior used to determine the user ID when LDAP.UID and LDAP.DOMAIN are defined - see LDAP.DOMAIN.

Option	Description
LDAP.AUTHENTICATION.MODE	(Optional) The method of authentication used to login into the LDAP server. You can choose from: (simple) - clear-text password authentication (none) - anonymous authentication The default is (simple).
LDAP.PWD	(Optional) The password used to login into the LDAP server.
LDAP.TIMEOUT	(Optional) The amount of time (in milliseconds) after which a connection attempt or query should expire. The default is 10000 (10 seconds).
LDAP.SEARCH.BASE	(Optional) The base of the search in the DIT. This is the starting point (node location) of a search in the DIT. If a value is not provided, the system looks for the LDAP.DOMAIN option and builds a search base from it.
LDAP.DOMAIN	(Optional) This is the domain of the LDAP server. It is used to build the user ID for login into the LDAP server by appending the at symbol (@) plus the value of this option to the LDAP.UID value. The value of LDAP.DOMAIN is further parsed into domain components which are used as the default value for LDAP.SEARCH.BASE, if not already defined.
LDAP.OBJECTS	(Optional) A semicolon-delimited filter list of object classes to search in the LDAP server. If defined, it overrides the default filter list of object classes to search: group and groupOfNames.

LDAP.OBJECTS.SEARCH.STRING	(Optional) An explicit string value to be used as the filter of object classes to search. If defined, it overrides any value provided for LDAP.OBJECTS option.
	The value provided for this option must be specified in the appropriate LDAP protocol filter format. Also, if the search filter contains a question mark (?), the system replaces it with the user ID passed in as an argument to this function. Here are some examples:
	`(|(objectClass=group)(objectClass=groupOfNames)).`
	`Cn=?`

LDAP.OBJECT.ATTRIBUTES	(Optional) The name of the attributes to retrieve for each object class, which contain a value that will be used to determine a match for USERID specified. The default values are *member* and *cn* (*cn* is always included).

Option	Description
LDAP.MATCH.ATTRIBUTES	(Optional) The name of one or more attributes that are contained within the value returned by a search for the LDAP.OBJECT.ATTRIBUTES option. This is the name of an attribute whose value will be used to compare vs. the USERID specified to determine a match. For example, if LDAP.OBJECTS contains a value of 'groupOfUniqueNames' and LDAP.OBJECT.ATTRIBUTES contains a value of 'uniqueMember' and value returned for the 'uniqueMember' attribute of 'groupOfUniqueNames' object class is 'uid=admin,ou=people, dc=mycompany,dc=com' and you want to match the USERID value with the value for 'uid', you would supply a value of 'uid' for this option. The default is *cn*.
LDAP.SEARCH.SCOPE	(Optional) The scope of the search. You can choose from: (base) - search only the named context (one) - search one level below the named context but not the named context (sub) - search the entire subtree, including the named context. The default is (sub).
LDAP.DEREF.LINK	(Optional) Enter No if you do not want the system to reference links to other nodes during a search. The default is Yes.
LDAP.VERSION	(Optional) An integer value indicating the LDAP protocol version to use. You can choose from: (2) - Version 2 (3) - Version 3 The default is (3).
LDAP.SEARCH.LEVEL	(Optional) This property specifies the search level. You can choose from 1 (USER) 2 (GROUPS) The system executes different logic to search group type objects or user type objects based on the search level specified. The default is 1 (USER).

Option	Description

LDAP.DN.IDENTIFIER	(Optional) The value for this property is used in the following ways:
	- In cases were LDAP.SEARCH.LEVEL is equal to 1 (USER) and there is no LDAP.OBJECTS.SEARCH.STRING value specified, the system generates a default search filter of the format identifier=userID, where *identifier* is the value of this property and *userID* is the user ID passed in as an argument to this function.
	- In cases were LDAP.SEARCH.LEVEL is equal to 2 (GROUPS) and there is no LDAP.OBJECTS.SEARCH.STRING value specified, the system generates a default search filter from LDAP.OBJECTS and LDAP.OBJECT.ATTRIBUTES, where each attribute value in the search filter is an asterisk (*). This tells the system to match any value for the attributes specified. If LDAP.RDNDS property is also provided, the asterisk (*) is replaced with identifer=userID, followed by a comma and the LDAP.RDNS value to fine tune the search, where *identifier* is the value for this property and *userID* is the user ID passed in as an argument to this function. Here is an example of a default search filter:
	`(&((objectClass=groupOfNames)(member=*)))`
	In a case where a value of
	`'CN=Users,DC=PDDC,DC=DOCUCORP,DC=COM'`
	is specified for LDAP.RDNS and this property contains a value of *CN*, the search filter generated would look like this:
	`(&((objectClass=groupOfNames)(member=CN=Administrator, CN=Users,DC=PDDC,DC=DOCUCORP,DC=COM))).`
	The default is *CN*.

Option	Description

LDAP.RDNS	(Optional) This property is only used when LDAP.SEARCH.LEVEL is equal to 2 (GROUPS) and when the LDAP.OBJECTS.SEARCH.STRING option is not specified. In such a case, the system builds a default search filter from LDAP.OBJECTS and LDAP.OBJECT.ATTRIBUTES and attribute values specified in the default search filter will contain an asterisk (*). This tells the system to match any value for the attributes specified.
	When this property is specified, the system uses the value along with the value for LDAP.DN.IDENTIFIER to replace the asterisk (*) and narrow the search, thereby speeding the process. Here is an example of a default search filter:
	`(&((objectClass=groupOfNames)(member=*)))`
	In a case where a value of
	`'CN=Users,DC=PDDC,DC=DOCUCORP,DC=COM'`
	is specified for this property and LDAP.DN.IDENTIFIER contains a value of *CN*, the search filter generated would look like this:
	`(&((objectClass=groupOfNames)(member=CN=Administrator, CN=Users,DC=PDDC,DC=DOCUCORP,DC=COM))).`

LDAP.USE.SSL	(Optional) A value of Yes enables encrypted communication through an SSL channel. For SSL connections to work, the LDAP server must be configured for SSL with a certificate from a trusted certification authority. This configuration is vendor specific, consult your vendor documentation for more information.
LDAP.DEBUG	(Optional) A value of Yes enables logging of debugging information to a file named *trace*.

SSL Configuration

These additional SSL input options are only supported in the Java rule:

Input variables

Variable	Description
LDAP.SSL.SOCKET.FACTORY.CLASS	(Optional) The name of a custom SSL socket factory class that should be used to override the default SSL socket factory used by JNDI - javax.net.ssl.SSLSocketFactory. This value should contain the package and class name of an SSL socket factory class that extends the javax.net.ssl.SSLSocketFactory class. There is no default value for this option. If this option is not specified and SSL support is enabled through LDAP.USE.SSL option, JNDI will use the javax.net.ssl.SSLSocketFactory class and look for the Java key and trust stores in the following way: Look for them in system properties javax.net.ssl.keyStore and javax.net.ssl.trustStore and look for their passwords in system properties javax.net.ssl.keyStorePassword and javax.net.ssl.trustStorePassword. If the system properties are not defined, look in the default key store/trust store named *cacerts* located in JAVA_HOME\jre\lib\security directory and use the default password (changeit) for them. Implementations that need to load their own key store and trust store and do not want to use the system properties can define their own SSLSocketFactory class or use the com.docucorp.util.LDAPSSLSocketFactory class in DocucorpUtil.jar package by providing the appropriate value for this option. See also the LDAP.USE.SSL, LDAP.SSL.PROTOCOL, LDAP.SSL.KEY.STORE, LDAP.SSL.KEY.STORE.TYPE, LDAP.SSL.KEY.STORE.MANAGER.TYPE, LDAP.SSL.KEY.STORE.PWD, LDAP.SSL.TRUST.STORE, LDAP.SSL.TRUST.STORE.TYPE, LDAP.SSL.TRUST.STORE.MANAGER.TYPE, and LDAP.SSL.TRUST.STORE.PWD properties.
LDAP.SSL.PROTOCOL	(Optional) The SSL protocol to use with a custom SSL socket factory class. The default is SSLv3. This is only used when SSL support is enabled through LDAP.USE.SSL option and when the LDAP.SSL.SOCKET.FACTORY.CLASS option is used with a value of 'com.docucorp.util.LDAPSSLSocketFactory'.
LDAP.SSL.KEY.STORE	(Optional) The path and file name of the Java key store where the private keys and public certificates are stored. This is only used when SSL support is enabled through LDAP.USE.SSL option and when the LDAP.SSL.SOCKET.FACTORY.CLASS option is used with a value of 'com.docucorp.util.LDAPSSLSocketFactory'.
LDAP.SSL.KEY.STORE.TYPE	(Optional) The type of key store used. The default is 'JKS' for Java Key Store. This is only used when SSL support is enabled through LDAP.USE.SSL option and when the LDAP.SSL.SOCKET.FACTORY.CLASS option is used with a value of 'com.docucorp.util.LDAPSSLSocketFactory'.
LDAP.SSL.KEY.STORE.MANAGER.TYPE	(Optional) The key store manager type. The default is 'SunX509'. This is only used when SSL support is enabled through LDAP.USE.SSL option and when the LDAP.SSL.SOCKET.FACTORY.CLASS option is used with a value of 'com.docucorp.util.LDAPSSLSocketFactory'.
LDAP.SSL.KEY.STORE.PWD	(Optional) The password for the SSL key store. This is only used when SSL support is enabled through LDAP.USE.SSL option and when the LDAP.SSL.SOCKET.FACTORY.CLASS option is used with a value of 'com.docucorp.util.LDAPSSLSocketFactory'.

Variable	Description
LDAP.SSL.TRUST.STORE	(Optional) The path and file name of the Java trust store where the trusted public certificates are stored. This is only used when SSL support is enabled through LDAP.USE.SSL option and when the LDAP.SSL.SOCKET.FACTORY.CLASS option is used with a value of 'com.docucorp.util.LDAPSSLSocketFactory'.
LDAP.SSL.TRUST.STORE.TYPE	(Optional) The type of trust store used. The default is 'JKS' for Java Key Store. This is only used when SSL support is enabled through LDAP.USE.SSL option and when the LDAP.SSL.SOCKET.FACTORY.CLASS option is used with a value of 'com.docucorp.util.LDAPSSLSocketFactory'.
LDAP.SSL.TRUST.STORE.MANAGER.TYPE	(Optional) The trust manager type. The default is 'SunX509'. This is only used when SSL support is enabled through LDAP.USE.SSL option and when the LDAP.SSL.SOCKET.FACTORY.CLASS option is used with a value of 'com.docucorp.util.LDAPSSLSocketFactory'.
LDAP.SSL.TRUST.STORE.PWD	(Optional) The password for the SSL trust store. This is only used when SSL support is enabled through LDAP.USE.SSL option and when the LDAP.SSL.SOCKET.FACTORY.CLASS option is used with a value of 'com.docucorp.util.LDAPSSLSocketFactory'.
LDAP.DEBUG	(Optional) A value of Yes enables tracing of debug information to a file. The name of the file depends on the value in the configuration file for log4j. The default file names for a log4j configuration are logconf.xml and utilconf.xml.

Output variables

Variable	Description
RESULTS	Success or failure. No matches means failure.</td></tr>
LDAPERROR	A standard LDAP error is returned as a rowset in case of failure. In such a case, the LDAPERROR will also be added as an entry in the ERRORS rowset.
LDAP.ENTRIES	Matches for the search criteria specified will be returned in the form of an LDAP.ENTRIES rowset, with each element named as an ENTRY in the rowset.

Example

Here is an example of a properties file:

```
ldap.uid=Administrator
ldap.pwd=~Encrypted 2XAUnkxUYlx7i5AnQ4m4E1m00
ldap.host=PDDC.pd.com
ldap.port=389
ldap.authentication.mode=simple
ldap.domain=PDDC.pd.com
ldap.objects.search.string=(&(objectClass=group)(cn=Administrators)
)
ldap.object.attributes=member
ldap.match.attributes=cn
ldap.debug=yes
```

Here is another example of a properties file:

```
ldap.user=uid=admin,ou=people,dc=mycompany,dc=com
```

```
ldap.pwd=~Encrypted 2XAUnkxUYlx7i5AnQ4m4E1m00
ldap.host=localhost
ldap.port=636
ldap.authentication.mode=simple
ldap.search.base=ou=roles,dc=mycompany,dc=com
ldap.objects=group;groupOfNames;groupOfUniqueNames
ldap.object.attributes=uniqueMember;member
ldap.match.attributes=uid;cn
ldap.debug=yes
ldap.version=3
ldap.search.scope=sub
ldap.deref.link=true
ldap.use.ssl=Y
ldap.ssl.protocol=SSLv3
ldap.ssl.socketFactory.class=com.docucorp.util.LDAPSSLSocketFactory
ldap.ssl.key.store=c:/docserv/keystore/javakeystore
ldap.ssl.key.store.pwd=~Encrypted 2yQgqaRIZkRJd6m8L7WWD1000
ldap.ssl.key.store.type=JKS
ldap.ssl.key.store.manager.type=SunX509
ldap.ssl.trust.store=c:/docserv/keystore/javakeystore
ldap.ssl.trust.store.pwd=~Encrypted 2yQgqaRIZkRJd6m8L7WWD1000
ldap.ssl.trust.store.type=JKS
ldap.ssl.trust.store.manager.type=SunX509
```

Here is another example of a properties file:

```
ldap.host=localhost
ldap.port=389
ldap.authentication.mode=none
ldap.search.base=ou=roles,dc=mycompany,dc=com
ldap.objects=group;groupOfNames;groupOfUniqueNames
ldap.object.attributes=uniqueMember;member
ldap.match.attributes=uid;cn
ldap.debug=yes
ldap.version=3
ldap.search.scope=sub
ldap.deref.link=true
```

Here is an example request type for the Java rule:

```
<section name="ReqType:TEST_LDAP_Search">
    <entry name="function">atcw32-&gt;ATCLoadAttachment</entry>
    <entry name="function">atcw32-&gt;ATCUnloadAttachment</entry>
    <entry name="function">java;com.docucorp.ids.rules.
        LDAPRule;LDAPS;transaction;search;ARG,RUNMSG=4</entry>
</section>
```

Here is an example request type for the C rule:

```
<section name="ReqType:TEST_LDAP_Search_2">
    <entry name="function">atcw32-&gt;ATCLoadAttachment</entry>
    <entry name="function">atcw32-&gt;ATCUnloadAttachment</entry>
    <entry name="function">dprw32-&gt;DPRSetConfig</entry>
    <entry name="function">dprw32-&gt;DPRSearchLDAP,
        RUNMSG=4</entry>
</section>
```

Keep in mind...

- Encrypted option values should be preceded by this keyword:

  ```
  ~Encrypted
  ```

  followed by a space (see the ldap.pwd value in the examples of a properties file). Values should be encrypted with the cryrun program or DataCrypt class in DocuCorpUtil package.

- The options in an INI file for a configuration available to IDS should be placed in a control group named LDAP. You must also provide a CONFIG input message variable or rule argument so IDS can find the LDAP control group in the appropriate INI file. Here is an example:

  The DAP.INI file configuration:

  ```
  < Config:Example >
      INIFile = example.ini
  ```

  The EXAMPLE.INI file configuration:

  ```
  < LDAP >
      ldap.host = localhost
      ldap.port = 389
      ldap.timeout = 10000
      ldap.uid = userID@PDDC.pd.com
      ldap.pwd = 123456xxx
      ldap.objects.search.string = cn=?
      ldap.authentication.mode = simple
      ldap.domain = PDDC.pd.com
      ldap.dn.identifier = cn
  ```

  The input message variable that is part of the request:

  ```
  CONFIG=Example
  ```

  The request type:

  ```
  <section name="ReqType:SearchLDAP">
  <entry name="function">atcw32-&gt;ATCLoadAttachment</entry>
  <entry name="function">atcw32-&gt;ATCUnloadAttachment</entry>
  <entry name="function">dprw32-&gt;DPRSetConfig</entry>
  <entry name="function">dprw32-&gt;DPRSearchLDAP>/entry>
  </section>
  ```

- If you are using jvm 1.3, you must replace the jsse.jar with one from jvm 1.4 at location JAVA_HOME\jre\lib\ext.

- Configuring the C rule with SSL involves installing the certificate submitted by the LDAP server into the trusted certification authorities store of the box where IDS is running. If the client program (IDS) is also to submit a certificate during the SSL hand-shake, then that certificate also needs to be installed into the trusted certification authorities store of the LDAP server.

- Configuring the Java rule with SSL involves importing the certificate submitted by the LDAP server into the Java trust store. If the client application (IDS) is also to submit a certificate during the SSL handshake, then that certificate also needs to be installed into the trusted certification authorities store of the LDAP server.

1782
IDS

USING THE NEW LDAP API

IDS now offers a new LDAP API for Java. The JAVA DocucorpUtil package now includes an LDAP class which you can use to query an LDAP server for group information for a user.

For more information please see the LDAP.html documentation that ships with IDS located in the dsi_sdk\java\docs\com\docucorp\util directory and see the ldapTest class example which ships with IDS and is located in the dsi_sdk\java\samples\ldap directory.

NOTE: If you are using JVM version 1.3, you must replace the jsse.jar file with the one from JVM version 1.4, which you can find at this location:

JAVA_HOME\jre\lib\ext

2198
IDS

MONITORING AND MANAGING IDS INSTANCES

You can use the new Watchdog process to manage and monitor IDS instances. The Watchdog process is the one that is now started, stopped, and configured as a service and it in turn is responsible for managing and monitoring the IDS instances. It monitors the health of each instance and restarts it or stops it when needed. You can also configure the watchdog process through log4j to send email notifications when an instance encounters a fatal or mission-critical error.

Watchdog also monitors the idle time for each instance and starts additional ones when all running instances are under load.

These options are supported:

Options	Description
Instances	(Optional) The number of IDS instances Watchdog should start at startup. The default is two (2).
UseLoadBalancing	(Optional) This option controls whether Watchdog checks the idle time of the instances that are running and starts additional ones when all of them are busy. Instances are considered busy when their idle time is less than the value provided in the MinIdleTimeSeconds option. Watchdog uses the value provided in the IdleTimeChecks option to determine the number of idle time checks to run before it starts additional instances. When additional instances are started for load balancing purposes, they are shut down by Watchdog if their idle time exceeds the value in the MaxIdleTimeSeconds option. The maximum number of instances running is the value for the ***MaxInstances*** option (including the instances configured in the ***Instances*** option). Watchdog checks the idle time of the current instances at the interval specified in the IdleTimeCheckIntervalSeconds and if all are busy, it starts an additional number of instances equal to the value provided in the IncrementCount option. Please note that Watchdog does not start checking the busy time of the current instances until the time provided in the IdleTimeCheckDelaySeconds option is reached. Make sure the value for the delay is ample enough to provide for all instances to start and reach an idle time equal to or greater than the value provided for the MinIdleTimeSeconds option. You can enter Yes (or True) or No (of False). The default is Yes.
MaxInstances	(Optional) This option controls the maximum number of instances that can run when the UseLoadBalancing option is enabled. The default is the number of processors times two.
IncrementCount	(Optional) This option controls how many additional instances are started during the current check when all instances running are busy and the UseLoadBalancing option is enabled. The default is two (2).
IdleTimeCheckIntervalSeconds	(Optional) This option controls how often Watchdog checks the idle time of the instances that are running to determine if they are busy so it can start additional ones when the UseLoadBalancing option is enabled. The default is 60 seconds.
IdleTimeCheckDelaySeconds	(Optional) This option controls the initial delay before the first idle time check is performed by Watchdog when the UseLoadBalancing option is enabled. This time should be ample enough to allow all instances to start and reach an idle time equal to or greater than the value provided for the MinIdleTimeSeconds option. The default is 120 seconds.

Options	Description
IdleTimeChecks	(Optional) This option defines the number of consecutive Idle time checks that must fail, meaning all instances were busy during each check, before more instances are started when the UseLoadBalancing option is enabled. Each check takes place at the IdleTimeCheckIntervalSeconds interval. The default is two (2).
MinIdleTimeSeconds	(Optional) This option controls the minimum idle time for each instance. The idle time represents how long it has been since an IDS instance processed the last request. If Watchdog detects an instance has an idle time less than the value provided for this option, it considers it busy for the purpose of load balancing. The default is five seconds.
MaxIdleTimeSeconds	(Optional) This option controls the maximum idle time for an additional instance. The idle time represents how long it has been since an IDS instance processed the last request. If Watchdog detects an instance which was started for the purpose of load balancing has reached an idle time greater than the value provided for this option, it sends the instance a shutdown request. The default is 120 seconds.
MaxTransactions	(Optional) This option controls the maximum number of transactions an instance can process before it is restarted by Watchdog. Enter **-1** to disable this option. The default is 10000.
MaxReportIntervalSeconds	(Optional) This option controls the maximum time interval that can elapse without an instance reporting back to Watchdog before it is restarted. The default is 120 seconds.
MaxUpTimeSeconds	(Optional) This option controls the maximum time interval an instance can run before it is restarted by Watchdog. Enter **-1** to disable this option. The default is 28800 seconds (8 hours).
MaxRestarts	(Optional) This option controls the maximum number of restart attempts that can occur within a time interval specified by the RestartIntervalSeconds option before Watchdog shuts down. Use this option to prevent Watchdog from attempting to restart instances infinite times when they cannot be started due to configuration errors and so on. The default is five restarts.
RestartIntervalSeconds	(Optional) This option controls the interval used with the MaxRestarts option to determine if Watchdog is having a problem starting instances and to prevent continuous or infinite restart attempts. The default is 60 seconds.

Options	Description
MaxMemoryUsagePercent	(Optional) This option controls the maximum percentage of the total JVM memory that can be used by an instance before Watchdog will restart it. Note that the total memory used in this calculation does not include any memory used by native code. This option is used with the MemoryChecks option. The default is 95.
MemoryChecks	(Optional) This option controls the total count of consecutive memory checks that must be present, where the memory usage by an instance exceeds the value provided for the MaxMemoryUsagePercent option for each check, at which point Watchdog will restart it. The interval for each memory check is controlled by the CheckIntervalSeconds option. The default is -1, which disables this option.
CheckIntervalSeconds	(Optional) This option controls the time interval used by Watchdog to check the health of each instance. The default is one (1) second.
UseJMX	(Optional) This option controls whether JMX is used to monitor additional health metrics for each instance. Enabling this option lets Watchdog also monitor class loading, memory usage, garbage collection, and deadlocks in Java code for each instance. Note that enabling this option requires an additional and separate TCP/IP port for each instance so that it can be started with a JMX agent. You can enter Yes (or True) or No (of False). The default is No. Only use this option for debugging or testing purposes. Do not use this option in production mode because it causes extra overhead and it requires additional ports be used.
JMXPort	(Optional) This option controls the starting JMX port to use when starting each instance with a JMX agent if the UseJMX option is enabled. Note that the starting port value should consider that each additional instance that is started will try to use a continuous/incremental port number. The default starting port value is 49163.
JMXCheckIntervalSeconds	(Optional) This option controls the time interval used to run JMX checks for each instance when the UseJMX option is enabled. The default is 60 seconds.
JMXMemoryChecks	(Optional) This option controls the total count of consecutive JMX memory checks that must be present, where the memory usage by an instance exceeds the value provided for the MaxMemoryUsagePercent option for each check, at which point Watchdog will restart it. The interval for each check is controlled by the JMXCheckIntervalSeconds option. The default is -1, which disables this option.

Options	Description
JMXVerboseMemory	(Optional) This option controls whether Watchdog turns on verbose memory to output GC statistics for each IDS instance when the UseJMX option is enabled. You can enter Yes (or True) or No (of False). The default is No.
JMXVerboseClassLoader	(Optional) This option controls whether Watchdog turns on verbose class loading for each IDS instance when the UseJMX option is enabled. You can enter Yes (or True) or No (of False). The default is No.
WaitForShutdownSeconds	(Optional) This option controls how long Watchdog waits for an instance to shut down after it issues a shutdown command and before it terminates the instance. The default is 20 seconds.
OrderedRestartIntervalSeconds	(Optional) This option controls the interval used for restarting each of the IDS instances in a sequential/ ordered manner when the MaxTransactions or MaxUpTime options are used. Watchdog restarts one instance at a time and waits for an amount of time equal to the value specified for this option before it restarts the next one and so on until it has restarted all of them. If you set this option to less than 60 seconds, you can negatively affect performance. The default is 60 seconds.

Here are the new Log4J categories and appenders used by Watchdog (see logconf.xml file included in the docserv directory):

- These mail categories and appenders are used to send email notifications during mission critical errors, such as when IDS has a fatal exception:

```
<!--Used by Watchdog to send email notifications.-->
<category name="EMAIL" additivity="false">
<priority value="ERROR"/>
<appender-ref ref="EMAIL"/>
</category>

<appender class="org.apache.log4j.net.SMTPAppender" name="EMAIL">
<param value="1" name="BufferSize"/>
<param value="10.1.20.148" name="SMTPHost"/>
<!--Comment out the SMTPUsername and SMTPPassword parameters to skip
authentication.-->
<!--
<param value="" name="SMTPUsername"/>
<param value="" name="SMTPPassword"/>
-->
<param value="support@acme.com" name="From"/>
<!--Use a comma delimited string of email addresses for To, cc and
bcc.-->
<param value="user@acme.com,user@acme.com" name="To"/>
<param value="user@acme.com,user@acme.com" name="cc"/>
<param value="user@acme.com,user@acme.com" name="bcc"/>
<param value="Error Message" name="Subject"/>
<param value="ERROR" name="threshold"/>
```

```
<layout class="org.apache.log4j.PatternLayout">
<param name="ConversionPattern" value="%d-[%t]-%m\n"/>
</layout>
</appender>
```

- This category is used to log informational output by Watchdog:

```
<!--Used to log Watchdog informational output.-->
<category name="Watchdog.output" additivity="false">
<priority value="INFO"/>
<appender-ref ref="watchdog-stdout"/>
<appender-ref ref="watchdog-allroll"/>
</category>
```

- These categories and appenders are used to log debug and error messages by Watchdog. Change the Priority value to 'DEBUG' to log debug messages.

```
<!--Used to log Watchdog debug and error messages.-->
<category name="com.docucorp.watchdog.Watchdog" additivity="false">
<priority value="ERROR"/>
<appender-ref ref="watchdog-stdout"/>
<appender-ref ref="watchdog-allroll"/>
</category>

<!--Logs Watchdog messages to stdout.-->
<appender name="watchdog-stdout"
class="com.docucorp.util.logging.IDSConsoleAppender">
<layout class="org.apache.log4j.PatternLayout">
<param name="ConversionPattern" value="%d-[%t]-%m\n"/>
</layout>
</appender>

<!--Watchdog Appender.-->
<appender name="watchdog-allroll"
class="com.docucorp.util.logging.IDSFileAppender">
<param name="Append" value="true"/>
<param name="File" value="watchdog.log"/>
<param name="Encoding" value="ISO-8859-1"/>
<layout class="org.apache.log4j.PatternLayout">
<param name="ConversionPattern" value="%d-[%t]-%m\n"/>
</layout>
</appender>
```

- These categories and appenders are used by each Watchdog instance monitor thread to log information for each instance monitored separately. Change the Priority value to 'DEBUG' to log debug messages.

```
  <!--Used to log each thread's Instance Monitor debug and error
messages.-->
<category name="com.docucorp.watchdog.monitor.InstanceMonitor"
additivity="false">
<priority value="ERROR"/>
<appender-ref ref="instance-stdout"/>
<appender-ref ref="instance-allroll"/>
</category>

<!--Logs Instance Monitor thread messages to stdout.-->
```

```
 <appender name="instance-stdout"
class="com.docucorp.util.logging.IDSConsoleAppender">
 <layout class="org.apache.log4j.PatternLayout">
 <param name="ConversionPattern" value="%d-[%t]-%m\n"/>
 </layout>
 </appender>

 <!--Logs Instance Monitor thread messages to separate file(s).-->
 <appender name="instance-allroll"
class="com.docucorp.watchdog.util.WatchdogFileAppender">
 <param name="Append" value="true"/>
 <param name="File" value="~THREADID.log"/>
 <param name="Encoding" value="ISO-8859-1"/>
 <layout class="org.apache.log4j.PatternLayout">
 <param name="ConversionPattern" value="%d-[%t]-%m\n"/>
 </layout>
 </appender>
```

- These categories and appenders are used to log debug and error information for IPC (Inter-Process Communication) messages between Watchdog and the instances. Change the Priority value to 'DEBUG' to log debug messages.

```
 <!--Used to log IPCConnector debug and error messages.-->
 <category name="com.docucorp.watchdog.ipc.IPCConnector"
additivity="false">
 <priority value="ERROR"/>
 <appender-ref ref="connector-stdout"/>
 <appender-ref ref="connector-allroll"/>
 </category>

 <!--Logs IPCConnector debug and error messages to stdout.-->
 <appender name="connector-stdout"
class="com.docucorp.util.logging.IDSConsoleAppender">
 <layout class="org.apache.log4j.PatternLayout">
 <param name="ConversionPattern" value="%d-[%t]-%m\n"/>
 </layout>
 </appender>

 <!--Logs IPCConnector debug and error messages to a file.-->
 <appender name="connector-allroll"
class="com.docucorp.watchdog.util.WatchdogFileAppender">
 <param name="Append" value="true"/>
 <param name="File" value="~THREADID.log"/>
 <param name="Encoding" value="ISO-8859-1"/>
 <layout class="org.apache.log4j.PatternLayout">
 <param name="ConversionPattern" value="%d-[%t]-%m\n"/>
 </layout>
 </appender>
```

A watchdog configuration file can contain multiple sections, each with its own set of options. Here are some examples:

Watchdog section

Here is an example of the Watchdog section:

```
<configuration>
    <section name="Watchdog">
        <entry name="UseJMX">No</entry>
        <entry name="JMXCheckIntervalSeconds">60</entry>
```

```
            <entry name="JMXMemoryChecks">3</entry>
            <entry name="JMXVerboseMemory">Yes</entry>
            <entry name="JMXVerboseClassLoader">Yes</entry>
        </section>
        <section version="2.2" name="DocumentServer">
            <entry name="StartCommand">java</entry>
            <entry name="StartArguments">-Djava.endorsed.dirs=lib/
    endorsed -Xmx256m -Ddsimessage.debug=N -Dmarshaller.output=N -cp
    .;lib/DocucorpStartup.jar -Dids.configuration=docserv.xml -
    Dlogging.configuration=logconf.xml com.docucorp.startup.Startup
    com.docucorp.ids.DocumentServer</entry>
            <entry name="StartDirectory">.</entry>
            <entry name="Instances">2</entry>
            <entry name="UseLoadBalancing">Yes</entry>
            <entry name="MaxInstances">10</entry>
            <entry name="IncrementCount">2</entry>
            <entry name="IdleTimeCheckIntervalSeconds">60</entry>
            <entry name="IdleTimeCheckDelaySeconds">120</entry>
            <entry name="IdleTimeChecks">2</entry>
            <entry name="MinIdleTimeSeconds">5</entry>
            <entry name="MaxIdleTimeSeconds">120</entry>
            <entry name="MaxTransactions">10000</entry>
            <entry name="MaxReportIntervalSeconds">60</entry>
            <entry name="MaxUptimeSeconds">28800</entry>
            <entry name="MaxRestarts">5</entry>
            <entry name="RestartIntervalSeconds">60</entry>
            <entry name="MaxMemoryUsagePercent">95</entry>
            <entry name="MemoryChecks">3</entry>
            <entry name="CheckIntervalSeconds">1</entry>
            <entry name="UseJMX">No</entry>
            <entry name="JMXPort">49163</entry>
            <entry name="JMXCheckIntervalSeconds">60</entry>
            <entry name="JMXMemoryChecks">3</entry>
            <entry name="JMXVerboseMemory">Yes</entry>
            <entry name="JMXVerboseClassLoader">Yes</entry>
            <entry name="WaitForShutdownSeconds">20</entry>
            <entry name="OrderedRestartIntervalSeconds">60</entry>
        </section>
    </configuration>
```

These JVM options are supported:

Option	Description
-Dwatchdog.configuration	(Optional) The name of the XML configuration file for watchdog. The default is docserv.xml.
-Dlog4j.configuration	(Optional) The name of the XML configuration file for LOG4J. The default is logconf.xml.
-Dwatchdog.prefix	(Optional) A unique string that should be used as the prefix for all Watchdog files/locks generated on disk. Use this option when running more than one Watchdog instance from the same directory.

Here are some examples:

Scenario 1 A platform contains a single CPU and the default values are used for the Instances option and for load balancing.

In this case, the default value of Instances will be two (2) and the default value of MaxInstances will also be two (2) so no load balancing will occur.

Scenario 2 A platform contains four CPUs and the default values are used for the Instances option and for load balancing.

In this case the default value of Instances will be two (2) and the default value of MaxInstances will be 8. The default increment count will be two (2), the default minimum idle time will be 5 seconds, and the default maximum idle time will be 120 seconds.

Load balancing will occur and Watchdog will check the idle time of any running instances every 60 seconds. If each of the instances running has an idle time that is less than 5 seconds, Watchdog deems them all busy and starts two additional instances. Watchdog then continues on to the next check interval.

These steps are repeated during each check interval until the total number of instances running reaches eight. If any of the running instances were started for the purpose of load balancing and reach an idle time greater than 120 seconds, they are shut down by Watchdog.

Scenario 3 A platform contains four CPUs and the value for the Instances option is set to 20 and the default values are used for load balancing.

In this case the value for MaxInstances will be 8 but the value for the instances will be greater than the value for the maximum instances that can be reached during load balancing so no load balancing will occur.

2247
IDS

Passing Additional JVM Options to DSILIB

Version 2.2 DSILIB uses Java through JNI (Java Native Interface) and as such it creates a Java Virtual Machine (JVM) at runtime. DSILIB now lets you pass JVM options before the JVM is created, so you can fine-tune what is created.

For instance, you can now specify the size of memory for the JVM. This is helpful, for example, if you need to set memory higher to handle large files transmitted via the message bus (queue).

To pass JVM options, use the *dsi_extended_properties* environment variable. This environment variable should contain a comma-delimited list of additional JVM options to pass during creation of a JVM.

Here is an example of how you would set the environment variable from a command prompt:

Windows `set dsi_extended_properties=-Xmx256m,-Dlog4j.configuration=logclientconf.xml`

UNIX `export dsi_extended_properties=-Xmx256m,-Dlog4j.configuration=logclientconf.xml`

Examples of client-based applications that use DSILIB include:

- ASP pages using IDSASP.DLL
- JSP pages using IDSJSP.jar
- DSIJava.jar files, which use the old legacy C code (DSILIB)
- The DSICOTB.EXE, DSITEST.EXE, and DSIEX.EXE test programs

2248
IDS

Generating Debug Output for Client Requests

IDS now supports new log4j categories and appenders which you can use in a log4j client configuration file to produce debugging output for client requests.

The following new categories and appenders are now supported:

```
<category name="Receive-Message">
    <priority value="DEBUG"/>
    <appender-ref ref="receive-message"/>
</category>

<category name="Send-Message">
    <priority value="DEBUG"/>
    <appender-ref ref="send-message"/>
</category>

<appender class="com.docucorp.util.logging.IDSFileAppender"
name="receive-message">
    <param value="false" name="Append"/>
    <param value="client-receive.msg" name="File"/>
    <param value="true" name="Close"/>
    <param value="ISO-8859-1" name="Encoding"/>
    <layout class="org.apache.log4j.PatternLayout">
        <param value="%m" name="ConversionPattern"/>
    </layout>
</appender>

<appender class="com.docucorp.util.logging.IDSFileAppender"
name="send-message">
    <param value="false" name="Append"/>
    <param value="client-send.msg" name="File"/>
    <param value="true" name="Close"/>
    <param value="ISO-8859-1" name="Encoding"/>
    <layout class="org.apache.log4j.PatternLayout">
        <param value="%m" name="ConversionPattern"/>
    </layout>
</appender>
```

NOTE: See the logclientconf.xml file for an example.

2257
IDS

USING DEFAULT TIME-OUTS FOR DSILIB-BASED CLIENT APPLICATIONS

Now IDS can use default time-outs for DSILIB-based client applications. You set these default using the following new configuration entries in the docclient.xml file:

- DefaultTimeoutSeconds
- MaxTimeoutSeconds
- MinTimeoutSeconds

NOTE: Examples of client-based applications that benefit from this feature are ASP pages using IDSASP.DLL, JSP pages using IDSJSP.jar, and the test programs DSICOTB.EXE, DSITEST.EXE, and DSIEX.EXE.

For instance, suppose you have hundreds of web applications installed on a single IIS or Java server and all of these applications are talking to the same IDS setup. Suppose some of these web applications have large time-out values which are not suitable for production mode, such as values longer than a few minutes. In this scenario, a transaction that takes a long time can tie up one thread on the web server. Since the total number of threads in the web server is limited, this can affect other applications.

Using these new options, the system administrator can make sure that no matter what was specified as the time-out value, the actual time-out period is what the system administrator thinks it should be.

These new entries go under the DocumentClient section in the docclient.xml file. Here is an example:

```
<section name="DocumentClient">
    <entry name="DefaultTimeoutSeconds">45</entry>
    <entry name="MaxTimeoutSeconds">60</entry>
    <entry name="MinTimeoutSeconds">30</entry>
```

Entry	Description
DefaultTimeoutSeconds	Use this entry when DSILIB-based client applications, such as dsiex, dsitest, and dsicotb test programs, provide a time-out value of zero (0) to DSIGetQueueRec calls to wait for a response message. The default is 15 seconds.
MaxTimeoutSeconds	Use this entry to set the upper limit for the time-out value when waiting for a response message. If a time-out value is specified for DSIGetQueueRec calls and it is greater than MaxTimeoutSecondsvalue, the MaxTimeoutSeconds is used instead. There is no default.
MinTimeoutSeconds	Use this entry to set the lower limit for the time-out value when waiting for a response message. If a time-out value is specified for DSIGetQueueRec calls and it is less than MinTimeoutSeconds value, the MinTimeoutSeconds is used instead. There is no default.

NOTE: It is possible that Microsoft Server script execution time-out limits could be set lower than the values specified for this feature. In those instances, the values of the Microsoft Server script execution time-out limits would be used. Please consult your Microsoft documentation for more information.

2259
IDS

RUNNING TIMED REQUESTS

IDS now lets you run timed requests repeatedly or just in the primary instance. You can use the following new entry attributes for a timed request entry under the Timers section in docserv.xml file:

Entry attribute	Description

RepeatInterval	Enter true or yes (case sensitive) to tell IDS to convert the text value provided for the entry into seconds and run the timed entry at each interval specified. Here are some few examples.
	This timed section runs every 120 seconds:
	`<entry RepeatInterval="yes" name="SSS">00:01:60</entry>`
	This timed section runs every 3720 seconds:
	`<entry RepeatInterval="yes" name="SSS">01:01:60</entry>`
	This timed section runs every 90 seconds:
	`<entry RepeatInterval="yes" name="SSS">90</entry>`
	If more than one IDS instance is running, any timed sections configured with the RepeatInterval attribute are run at a random interval using the interval seconds as the seed. Making sure they are not run at the same time by all IDS instances, allows other processing to take place.
	The default lower bound is 60 seconds, meaning any timed section that is configured to use a time interval of less than 60 seconds will instead use the default value.

RunOnPrimaryInstanceOnly	Enter yes or true (case sensitive) to make sure only the primary instance runs the timed section. For instance, you might want to do this when a timed section runs a synchronization rule that updates resources for a master resource library. This type of request would only need to be run once. If you omit this attribute, all IDS instances will run each timed section. Here is an example: `<entry RunOnPrimaryInstanceOnly="Yes" name="SSS">00:01:60</entry>`

2269
IDS

USING THE NEW JAVA TEST UTILITY

IDS now includes a Java threads test utility you can run to send requests to IDS using single or multiple threads. The threads utility lets you send test messages to IDS. It can send one test message at a time or it can create multiple threads to send concurrent test messages to IDS. It also supports sending rowsets and attachments in a test message.

Additionally, it can process all messages contained in a receive.msg file that was generated by IDS by enabling the capturing of said file through the appropriate log4j options (see the ReceiveMessage category and receivemessage appender sections in the logconf.xml file for IDS). This is useful when you need to recreate an error in another IDS environment by running the same request messages on it.

You can invoke the test utility via the threads script (threads.bat) included with IDS.

NOTE: You must have Java version 5 or later installed to use this test utility. You must also have The latest DocuCorpUtil.jar and watchdog.jar files.

If you invoke the test utility without any arguments, it displays usage information.

Parameters

Here are the parameters you can use with the threads. bat script:

[properties|configuration|config]

(Required) The name of a properties or XML configuration file in the same format as a docclient.xml file containing the marshaller, queue factory, and message bus properties to use.

Here is an example of a properties file:

```
marshaller.class=com.docucorp.messaging.data.marshaller.SOAPMIMEDSI
MessageMarshaller
queuefactory.class=com.docucorp.messaging.http.DSIHTTPMessageQueueF
actory
http.url=http://localhost:49152
```

Here is an example of an xml configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<section name="DocumentClient">
<section name="messaging">
<section name="queue">
<entry
name="marshaller.class">com.docucorp.messaging.data.marshaller.SOAP
MIMEDSIMessageMarshaller</entry>
<entry
name="queuefactory.class">com.docucorp.messaging.http.DSIHTTPMessag
eQueueFactory</entry>
<entry name="http.url">http://localhost:49152</entry>
</section>
</section>
</section>
</configuration>
```

[message|msg]

The name of a message file you want the utility to use to compose a request message. The default is an SSS request message.

Here is an example of the new format; which supports messagevars, text files, binary files, and rowsets sections):

```
[ messagevars ]
requestype=sss
config=sampco
userid=docucorp

[ textfiles ]
ATTACHMENT1=test.xml

[ binaryfiles ]
ATTACHMENT2=test.pdf
```

```
[ rowsets ]
count=2
rowset.1.name=records
rowset.1.rows=2
rowset.1.row.names=a,b,c
rowset.1.row.1.values=value1,value2,value3
rowset.1.row.2.values=value1,value2,value3
rowset.2.name=tables
rowset.2.rows=1
rowset.2.row.names=table,location
rowset.2.row.1.values=wip,mstrres\amergen\wip
```

Here is an example of the old format, which just includes name/value pairs:

```
reqtype=sss
config=SAMPCO
```

[msgvars] Include additional, colon-delimited message variables from the command line that should be added to the request message. Here is an example:

```
threads msgvars=reqtype=sss:config=SAMPCO
```

[reqtype] Enter the request type present in the server configuration (docserv.xml) file. The default is SSS.

[transfile/receivefile] Enter the name of a file that contains one or more requests processed by IDS. For example, this could be a receive.msg file generated by IDS via the appropriate debug options.

This is useful when you are trying to recreate an error reported when the end user provides this file.

[buffer] Specify a file containing a SOAP message that you want placed directly into the queue. For example, this could be a receive.msg file generated by IDS via the appropriate debug option and which contains a single request that was processed by IDS.

This is useful when you are trying to recreate an error reported when the end user provides this file. This is also useful when you are testing request messages generated by third party applications.

[nothreads] Enter the number of threads you want to create. Use this parameter with the notrans parameter, where for each thread *N* in nothreads, notrans transactions are run. The default is one (1) thread.

[timeout] Enter the timeout value to use in milliseconds for each transaction. The default is 60000 ms (60 seconds).

[display] Enable this parameter to display the result message returned by IDS. The default is False.

[time] Enable this parameter to display the time for each thread's transaction and the total processing time for all transactions (all threads). The default is False.

[infinite] Use this parameter to specify whether to process the specified number of threads in an infinite loop. For instance, if you specify True, the loop continues to generate the specified number of threads after the interval expires. Use this parameter with the interval parameter. The default is False.

[interval] Enter, in milliseconds, an interval value in which the infinite loop will again spawn the number of threads specified. Use this parameter with the infinite parameter. The default is 1000 ms (1 second).

[notrans] This parameter specifies the total number of transactions to process for each thread. Use this value with the nothreads parameter, where for each thread *N* in nothreads, notrans transactions are run. The default is one (1).

[msgno] Use this parameter with the transfile or receivefile parameter. It should indicate a message number present in the transaction file specified, which is used to generate a request.

[range] Use this parameter with the transfile or receivefile parameter. It should specify a range of message numbers present in the transaction file specified, which are used to generate requests, such as 1,5.

[msglist] Use this parameter with the transfile or receivefile parameter. It should specify a comma-delimited list of message numbers present in the transaction file. These message numbers, such as 1,3,5,7, are then used to generate requests.

[unique] Enter True if you want the utility to generate a unique ID. The default is False, which means the message bus will generate the unique ID.

[noattchs] The number of file attachments to send per transaction. The utility looks for a file named *sendfiles.msg* when you specify this parameter. See the description of the sendfiles parameter for details about the format of this file. Use this parameter or the sendfiles parameter but not both.

You can also use the atcfile parameter when you are only sending one attachment. The utility prompts you for the attachment information if needs.

[sendfiles] Enter the name of the file that contains the file attachments to send for a transaction. Use this parameter or the noattchs parameter but not both.

You can also use the atcfile parameter instead when only sending one attachment. The utility prompts you for the attachment information it needs.

Here is an example file:

```
name=ATTACHMENT1
file=c:\docserv\test.xml
type=text
name=ATTACHMENT2
file=c:\docserv\test.pdf
type=binary
```

[atcfile] Include this parameter to tell the utility to send one file attachment as part of the transaction. The utility prompts you for the attachment information it needs.

Do not use the noattchs or sendfiles parameters with this parameter.

[norcvs] Enter the number of file attachments to receive per transaction. The utility looks for a file named *receivefiles.msg* when you use this parameter. See the description of the receivefiles parameter for details about the format of this file. You can use this parameter or the receivefiles parameter but not both.

You can also use the rcvfile parameter when you are only receiving one attachment. The utility prompts you for the attachment information it needs.

[receivefiles] Enter the name of the file that contains the file attachments to receive for a transaction. Use this parameter or the norcvs parameter but not both.

You can also use the rcvfile parameter instead when only receiving one attachment. The utility prompts you for the attachment information it needs.

Here is an example file:

```
name=ATTACHMENT1
file=c:\docserv\input-test.xml
name=ATTACHMENT2
file=c:\docserv\input-test.pdf
```

[rcvfile] Include this parameter to tell the utility to receive one file attachment as part of the transaction. The utility then prompts you for the attachment information it needs.

Do not use the norcvs or receivefiles parameters with this parameter.

[receivepath] Use this parameter to specify the default location to receive file attachments when the rcvfile, receivefiles, and norcvs parameters are not used. The default is the current directory.

Examples

Here are some Windows examples:

```
threads.bat config=docclient.xml reqtype=sss time=yes notrans=100

threads.bat config=docclient.xml msgvars=reqtype=tsttest:test=14
display=yes

threads.bat properties=dsimsg.properties message=sss.ini
nothreads=300 timeout=30000 display=true time=true infinite=true
interval=30000.
```

INI Options

Change the priority value to DEBUG for the com.docucorp.test.threads category in the logclientconf.xml file so the utility can output the location of any files it receives from IDS during testing.

2278
IDS

Using Client Connection Definition Tables

The MQSeries message bus can now read connection information from Client Connection Definition Table (CCDT) files. The code can then use any queue manager listed in the CCDT file to establish a connection.

NOTE: Support for CCDT files in Java requires WebSphere MQ, version 6.0 or later. Refer to the WebSphere MQ documentation for information about Client Connection Definition Tables.

For additional information, see the description of the mq.ccdt.url property in the HTML documentation for the com.docucorp.messaging.mqseries.DSIMQMessageQueueFactory class. This information is included with the Java SDK.

2306
IDS

WEBSPHERE MQ SECURITY EXIT SUPPORT

Now you can attach custom security exits to WebSphere MQ (MQSeries) queues.

Security exits are external libraries of code that can be installed and run in WebSphere MQ queues. For IDS, security exits consist of a Java class in a .jar file, with an optional native component.

To have a security exit installed and run, you need to know the name of the Java class for the security exit and the name of the .jar file that has the security exit.

In the docserv.xml configuration file, set up a queue section for WebSphere MQ queues. In that section, add an entry similar to the one shown here:

```
<entry name="mq.customsecurityexit.classname">com.customer.
    securityClassName<entry>
```

Substitute the name of the your security exit Java class name for:

```
mq.customsecurityexit.classname
```

You must load the .jar file that has the custom security exit code. For application servers running running Docupresentment client code, refer to the application server's documentation for information on modifying the classpath for the web application or for including a .jar file in a particular directory.

For Docupresentment server, you can either...

- Put the .jar file in the server's lib directory, or
- Modify how Docupresentment server is run by adding the System property

  ```
  com.skywiresoftware.extraClasspath
  ```

  with a reference to any .jar files needed to run the security exit.

For example, for the docserver.bat file, you could add an entry like the one shown here:

```
-Dcom.skywiresoftware.extraClasspath=/path/to/security.exit.jar
```