SYBASE®
An SAP® Company

System Administration

Sybase Mobiliser Platform 5.1 SP03

DOCUMENT ID: DC01970-01-0513-01

LAST REVISED: September 2013

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

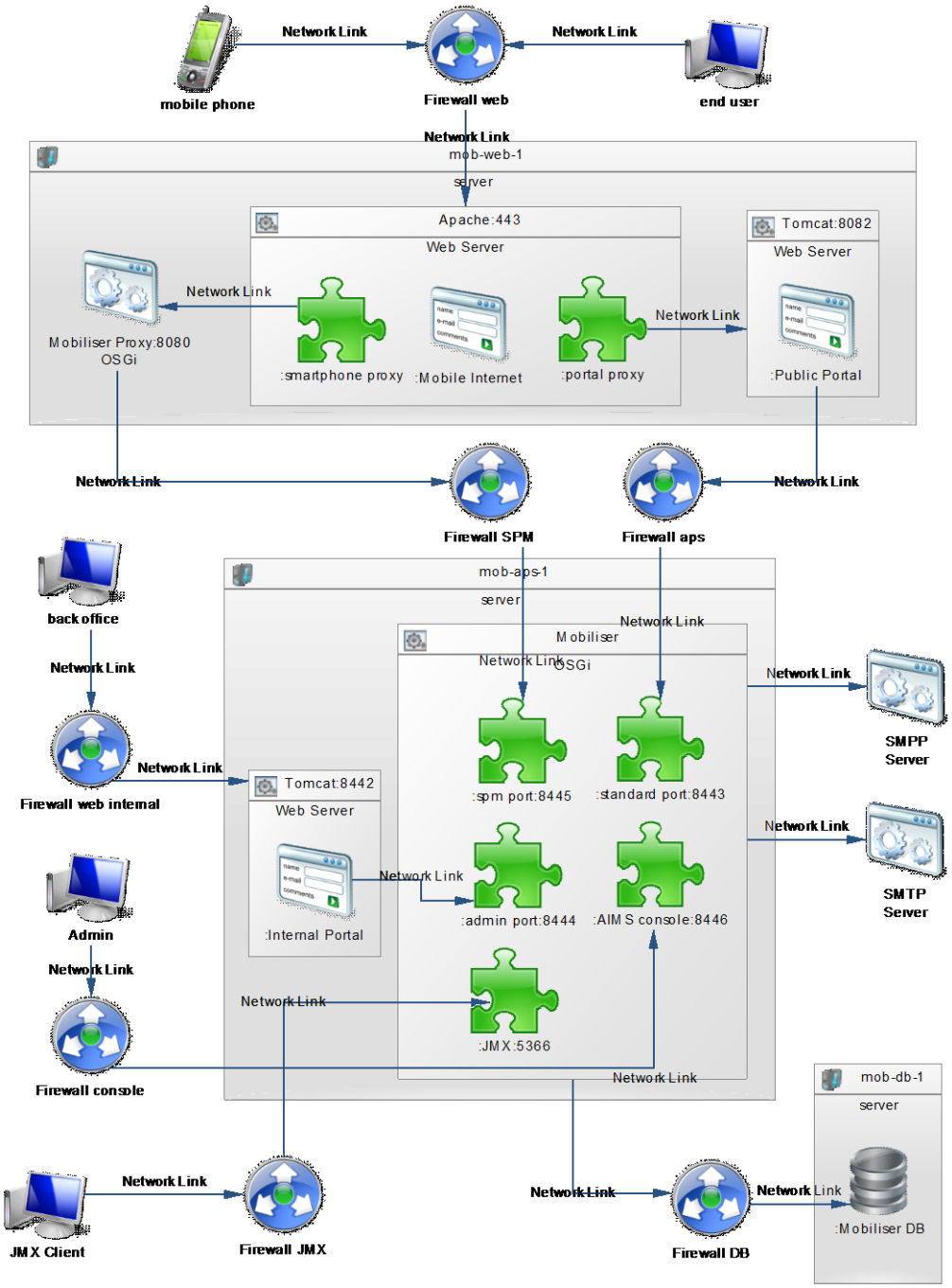
Standard Deployment Model

Each Sybase® Mobiliser Platform host must meet the requirements for operating system and available disk space. In a development or test environment, you can install the system on a single physical host or virtual machine. In a production environment, deploy the system in a tiered manner to aid in administration, maintenance, and security.

The standard Mobiliser Platform tiered architecture contains:

- Web layer (customer self-service portal)
- Application layer (Web service, back office)
- Database layer

Standard Deployment Model



System Reference

To manage the system effectively, it is crucial to know about Mobiliser Platform subsystem components and how they fit together. This part outlines many important aspects of the system in quick reference format.

It covers the location of crucial system files and file systems, as well as other reference material that you might need when you are administering Mobiliser Platform: for example, installation directories, port numbers, network ports, and port and host name configuration properties.

Installation Directories

The Mobiliser Platform software ZIP file contains everything you need to complete a successful installation. All Mobiliser Platform directories should reflect those outlined in the *Creating the Directory Structure and Copying Files* section in the *Sybase Mobiliser Platform Installation* guide.

According to the Standard Deployment Model for Mobiliser Platform, the entire platform is installed across three servers:

- mob-web-1
- mob-aps-1
- mob-db-1

The Mobiliser Platform installation directories should reflect the following on each server:

Table 1. Installation Directories (mob-web-1)

Object	File Path
Public Tomcat Container	/opt/sybase/portal/
Mobiliser Platform Proxy	/opt/sybase/proxy
Apache HTTPD Configuration	/opt/sybase/httpd
Smartphone Mobiliser Mobile Internet Version	/opt/sybase/mobileweb

Table 2. Installation Directories (mob-aps-1)

Object	File Path
Internal Tomcat Container	/opt/sybase/portal/
Mobiliser Platform Container	/opt/sybase/money

Table 3. Installation Directories (mob-db-1)

Object	File Path
Database Script Archives	/opt/sybase/db/sql

Port Number Reference

Components of Mobiliser Platform rely on communication ports for inter-process coordination, data transfer, and administrative access.

Change the component port numbers after installation, if necessary.

Proceed with caution when changing port numbers because the change might impact other configuration files that point to that port.

Table 4. Port Number References

Port	Description	Instructions for Changing
8080	The standard port for the Mobiliser Platform validation proxy.	**File:** /opt/sybase/proxy/conf/cfgbackup/com.sybase365.mobiliser.framework.service.proxy.properties **Server:** mob-web-1
8082	The default port utilized by the Tomcat instance hosting the external Web portal.	**File:** /opt/sybase/portal/conf/server.xml **Server:** mob-web-1
443	The default secure port used by Apache httpd, where external traffic reaches the mob-web-1 server.	**File:** /etc/httpd/conf/httpd.conf **Server:** mob-web-1
8445	The default listening port for the Mobiliser Smartphone endpoint.	**File:** /opt/sybase/money/conf/jetty.xml **Server:** mob-aps-1
8443	The default listening port for the Mobiliser Consumer Portal endpoint.	**File:** /opt/sybase/money/conf/jetty.xml **Server:** mob-aps-1
8444	The default listening port for the Mobiliser Administrative Portal endpoint.	**File:** /opt/sybase/money/conf/jetty.xml **Server:** mob-aps-1
8446	The default listening port for the AIMS Administrative Console.	**File:** /opt/sybase/money/conf/jetty.xml **Server:** mob-aps-1
5366	The default listening port for the Java Management Extension (JMX) Administrative console.	**File:** /opt/sybase/money/conf/cfgbackup/ com.sybase365.mobiliser.framework.gateway.security.authentication.jmx.properties **Server:** mob-aps-1
8442	The default port utilized by the Tomcat instance hosting the internal Web portal.	**File:** /opt/sybase/portal/conf/server.xml **Server:** mob-aps-1

Network Ports

Configure your firewalls to allow communication between the different Mobiliser Platform nodes. The Standard Deployment Model diagram on page 1 illustrates the network ports.

This table describes the default port configuration of Mobiliser Platform. For more details on how to change the ports used by Mobiliser Platform, see *Port and Host Name Configuration* on page 5.

Table 5. Network Ports

Name	Source	Destination	Protocol
PROXY2MOB	mob-web-1:*	mob-aps-1:8445	HTTPS
EUSER2MOB	mob-web-1:*	mob-aps-1:8443	HTTPS
JMX2MOB	Internal_JMX:*	mob-aps-1:5366	HTTPS
ADMIN2MOB	Admin_WS:*	mob-aps-1:8446	HTTPS
BO2INTTOMCAT	Backoffice_WS:*	mob-aps-1:8442	HTTPS
MOB2DB	mob-aps-1:*	mob-db-1:<db_listener>	HTTP
ALL2WEB	*	mob-web-1:443	HTTPS
WEB2PROXY	mob-web-1:*	mob-web-1:8080	HTTP
WEB2TOMCAT	mob-web-1:*	mob-web-1:8082	HTTP

Note: During installation, files are copied between servers using `scp` command. Therefore, access to port 22 from mob-aps-1 to mob-web-1 and mob-db-1 is required. If port 22 in your installation is unavailable, use an alternative method for copying the files onto the target machines.

Port and Host Name Configuration

You can change your host names and ports after installation. The subsections refer to the network connection in correspondence to Table 5. Network Ports on page 5.

PROXY2MOB

Source Configuration

The source configuration validates the proxy container that is installed on the mob-web-1 server. The configuration for the host to which the incoming requests are forwarded after validation is at:

```
/opt/sybase/proxy/conf/cfgbackup/com.sybase365.mobiliser.framework.service.proxy.properties
```

```
prefix=/mobiliser
```

```
scheme=https
host=mob-aps-1
port=8445
path=/mobiliser
```

Destination Configuration

The destination configuration must be changed in two files to change the default listening port of 8445. The configuration files are:

- /opt/sybase/money/conf/jetty.xml (on mob-aps-1)

```
<Call name="addConnector">
    <Arg>
      <New
class="org.eclipse.jetty.server.ssl.SslSelectChannelConnector
">
        <Arg><Ref id="sslContextFactory" /></Arg>
        <Set name="Port">8445</Set>
        <Set name="maxIdleTime">30000</Set>
        <Set name="Acceptors">2</Set>
        <Set name="AcceptQueueSize">100</Set>
        <Set name="statsOn">false</Set>
        <Set name="lowResourcesConnections">20000</Set>
        <Set name="lowResourcesMaxIdleTime">5000</Set>
        <Set name="forwarded">true</Set>
      </New>
    </Arg>
  </Call>
```

- /opt/sybase/money/conf/system.properties (on mob-aps-1)

```
# assign smartphone endpoints to their own port
com.sybase365.mobiliser.framework.gateway.smartphone.ports=80
85,8445
com.sybase365.mobiliser.framework.gateway.spmmbanking.ports=8
085,8445
```

Note: If the application server host name is changed in the installation environment from the default (mob-aps-1) in ANY Mobiliser Platform configuration file, the Jetty key must be recreated to reflect the new host name. For more information, see *Creating the Jetty Key* section in the *Sybase Mobiliser Platform Installation*.

EUSER2MOB

Source Configuration

The source configuration file is:

/opt/sybase/portal/bin/setenv.sh (on mob-web-1)

```
# change this value to connect to a different Mobiliser
hostname or port
MOBILISER_HOST=https://mob-aps-1:8443
```

Destination Configuration

The destination configuration file is:

/opt/sybase/money/conf/jetty.xml (on mob-aps-1)

```
      <Call name="addConnector">
      <Arg>
        <New class="org.eclipse.jetty.server.ssl.
SslSelectChannelConnector">
          <Arg><Ref id="sslContextFactory" /></Arg>
          <Set name="Port">8443</Set>
          <Set name="maxIdleTime">30000</Set>
          <Set name="Acceptors">2</Set>
          <Set name="AcceptQueueSize">100</Set>
          <Set name="statsOn">false</Set>
          <Set name="lowResourcesConnections">20000</Set>
          <Set name="lowResourcesMaxIdleTime">5000</Set>
          <Set name="forwarded">true</Set>
        </New>
      </Arg>
    </Call>
```

JMX2MOB

Source Configuration

The default configuration file of the installed JMX Console is directed to:

https://mob-aps-1:5366

Destination Configuration

The destination configuration file is:

/opt/sybase/money/conf/cfgbackup/
com.sybase365.mobiliser.framework.gateway.security.authentication.jmx.properties

```
# the url which clients will use to connect
```

```
# make sure this port matches the port above
serviceUrl=service:jmx:rmi://mob-aps-1:5366/jndi/rmi://mob-
aps-1:5366/jmxrmi
```

ADMIN2MOB

Source Configuration

A source configuration file does not exist and is not necessary, but the administrative workstation must have an Internet browser installed that can access the network and Mobiliser Platform endpoint hosting the AIMS console:

https://mob-aps-1:8446

Destination Configuration

The destination configuration must be changed in two files in order to change the default listening port of 8446.

The configuration files are:

- /opt/sybase/money/conf/jetty.xml

```
<Call name="addConnector">
    <Arg>
      <New class="org.eclipse.jetty.server.ssl.
SslSelectChannelConnector">
        <Arg><Ref id="sslContextFactory" /></Arg>
        <Set name="host">0.0.0.0</Set>
        <Set name="Port">8446</Set>
        <Set name="maxIdleTime">30000</Set>
        <Set name="Acceptors">2</Set>
        <Set name="AcceptQueueSize">100</Set>
        <Set name="statsOn">false</Set>
        <Set name="lowResourcesConnections">20000</Set>
        <Set name="lowResourcesMaxIdleTime">5000</Set>
        <Set name="forwarded">true</Set>
      </New>
    </Arg>
  </Call>
```

- /opt/sybase/money/conf/cfgbackup/com.sybase365.mobiliser.framework.gateway.security.authentication.webconsole.properties

```
requiredPorts=8086,8446
```

BO2INTTOMCAT

Source Configuration

A source configuration file does not exist and is not necessary, but the administrative workstation must have an Internet browser installed that can access the network and the Tomcat connector port on:

https://mob-aps-1:8442

Destination Configuration

The destination configuration file is:

/opt/sybase/portal/conf/server.xml (on mob-aps-1)

```
<Connector port="8442" protocol="HTTP/1.1" SSLEnabled="true"
maxThreads="150"scheme="https"secure="true"clientAuth="false"
sslProtocol="TLS"
keystoreFile="${catalina.home}/conf/keys/server/keystore"
keystorePass="Judson" />
```

MOB2DB

Source Configuration

Depending on the database credentials used, the source configuration files need to be adjusted in different areas.

If the database credentials are not changed, the only file that must be changed to specify the Java database connectivity (JDBC) URL used to reach the database is:

/opt/sybase/money/conf/system.properties (on mob-aps-1):

```
JDBC_URL=jdbc:sybase:Tds:localhost:5000/mobr5?APPLICATIONNAME
=mobr5&CACHE_COLUMN_METADATA=true
```

If the database credentials are altered from the default script (specified in the *Create the Database Schema (mobr5)* section in the *Sybase Mobiliser Platform Installation* guide), then you need to update the credentials in:

- /opt/sybase/money/conf/cfgbackup/ com.sybase365.mobiliser.framework.persistence.jdbc.bonecp.pool.properties (on mob-aps-1):

  ```
  username=mobr5
  password={enc}nsoVN/2Kv4askDeZiY+DH8KYDseo0Jd5C8CJNlKpGlA=
  ```

- /opt/sybase/money/conf/cfgbackup/ com.sybase365.mobiliser.util.report.crystalreports.properties (on mob-aps-1):

  ```
  jdbc.user=mobr5
  jdbc.password={enc}nsoVN/2Kv4askDeZiY+DH8KYDseo0Jd5C8CJNlKpGl
  A=
  ```

For security reasons, it is recommended that all database passwords be encrypted before being placed in any Mobiliser configuration file. For more details on how to encrypt configuration file passwords, see *Encryption of Configuration Files* on page 20.

Destination Configuration

The destination configuration file is the configuration of the database listener. In an ASE database the configuration for the database listening port by default is:

/opt/sybase/interface (on mob-db-1):

```
master tcp ether mob-db-1 5000
query tcp ether mob-db-1 5000
```

ALL2WEB

Source Configuration

The source configuration files come from a network/messaging hub, which is responsible for the configuration and routing.

Destination Configuration

The Apache HTTPD server configuration is the destination configuration, which is located in:

/etc/httpd/conf/httpd.conf (on mob-web-1)

```
Include /opt/sybase/httpd/conf/mobiliser_httpd_ssl.conf
```

The listening port is configured in:

/opt/sybase/httpd/conf/mobiliser_httpd_ssl.conf (on mob-web-1):

```
Listen 443
<VirtualHost *:443>
  SSLCertificateKeyFile   /opt/sybase/httpd/keys/server.key
  SSLCertificateFile      /opt/sybase/httpd/certs/server.crt
```

WEB2PROXY

Source Configuration

The source configuration is:

/opt/sybase/httpd/conf/mobiliser_httpd_locations.conf (on mob-web-01)

```
   # This is the standard URL for the mobiliser services (SOAP
and REST)

   ProxyPass /mobiliser/smartphone
http://localhost:8080/mobiliser/smartphone

   ProxyPassReverse /mobiliser/smartphone
http://localhost:8080/mobiliser/smartphone

   ProxyPass /mobiliser/rest/smartphone
http://localhost:8080/mobiliser/rest/smartphone

   ProxyPassReverse /mobiliser/rest/smartphone
http://localhost:8080/mobiliser/rest/smartphone
```

```
   # additional services to store/retrieve binary data are
located under mobiliser/binary

   ProxyPass /mobiliser/binary
http://localhost:8080/mobiliser/binary

   ProxyPassReverse /mobiliser/binary
http://localhost:8080/mobiliser/binary

   ProxyPass /mobiliser/rest/binary
http://localhost:8080/mobiliser/rest/binary

   ProxyPassReverse /mobiliser/rest/binary
http://localhost:8080/mobiliser/rest/binary
```

Destination Configuration

The source configuration validates the proxy container's jetty connector that is configured in:

/opt/sybase/proxy/conf/jetty.xml (on mob-web-1)

```
   <Call name="addConnector">
       <Arg>
         <New
   class="org.eclipse.jetty.server.nio.SelectChannelConnector">
           <Set name="host">0.0.0.0</Set>
           <Set name="port">8080</Set>
           <Set name="maxIdleTime">300000</Set>
           <Set name="Acceptors">2</Set>
           <Set name="statsOn">false</Set>
           <Set name="lowResourcesConnections">20000</Set>
           <Set name="lowResourcesMaxIdleTime">5000</Set>
           <Set name="forwarded">true</Set>
         </New>
       </Arg>
     </Call>
```

WEB2TOMCAT

Source Configuration

The source configuration is:

/opt/sybase/httpd/conf/mobiliser_httpd_locations.conf (on mob-web-01)

```
   # this forwards all /portal requests to the tomcat on
   localhost
ProxyPass /portal http://localhost:8082/portal

ProxyPassReverse /portal http://localhost:8082/portal
```

Destination Configuration

The Apache Tomcat container is configured to accept traffic from the portal proxy over port 8082. The destination configuration file is:

/opt/sybase/portal/conf/server.xml (on mob-web-01)

```
<Connector port="8082" protocol="HTTPS/1.1"
               connectionTimeout="20000"
               redirectPort="8442" />
```

INTTOMCAT2MOB

Source Configuration

The source configuration file is:

/opt/sybase/portal/bin/setenv.sh (on mob-aps-1)

```
# change this value to connect to a different Mobiliser
hostname or port
MOBILISER_HOST=https://mob-aps-1:8444
```

Destination Configuration

The destination configuration must be changed in two files in order to change the default listening port of 8444.

The configuration files are:

- /opt/sybase/money/conf/jetty.xml (on mob-aps-1)

```
<Call name="addConnector">
 <Arg>
   <New
class="org.eclipse.jetty.server.ssl.SslSelectChannelConnector">
     <Arg><Ref id="sslContextFactory" /></Arg>
     <Set name="host">0.0.0.0</Set>
     <Set name="Port">8444</Set>
     <Set name="maxIdleTime">30000</Set>
     <Set name="Acceptors">2</Set>
     <Set name="AcceptQueueSize">100</Set>
     <Set name="statsOn">false</Set>
     <Set name="lowResourcesConnections">20000</Set>
     <Set name="lowResourcesMaxIdleTime">5000</Set>
     <Set name="forwarded">true</Set>
   </New>
```

```
    </Arg>
  </Call>
```

- /opt/sybase/money/conf/system.properties on (mob-aps-1)

```
# assign management endpoints to their own port
com.sybase365.mobiliser.framework.gateway.management.ports=8084,8444
```

System Reference

Database User

In the standard deployment model, Mobiliser Platform database components are installed on the mob-db-1 server.

The standard installation scripts create a new user/schema (mobr5) with a standard password.

If you would like to choose a different user name and password for this user you should do so before executing the "`001_MONEY_drop_and_create_user.DDL`" script. The required steps are described below.

To change the password for the user, you can use the standard commands to change user passwords for the database of your choice. Remember to configure the changed password in the Mobiliser Platform configuration files to let Mobiliser know which password to use to connect to the database. The necessary changes are described in *MOB2DB* on page 9.

Change Database User Name and Password

The default Mobiliser Platform database schema is named "mobr5" with a password of "paybox" to access it. The mobr5 schema is created after manually running the `001_MONEY_drop_and_create_user.DDL` script for the supported database platform installed. You can change the name of the Mobiliser Platform schema before executing it by modifying the `001_MONEY_drop_and_create_user.DDL` file.

Table 6. Database Schema Changes contains the information for modifying the Mobiliser Platform database schema.

Table 6. Database Schema Changes

Database	Mobiliser Platform Schema Changes

ASE	`name = "<new_db_name>data"`
	`physname = "/opt/sybase/data/<new_db_name>.dat",`
	`name = "<new_db_name>log",`
	`physname = "/opt/sybase/data/<new_db_name>.log",`
	`sp_addlogin "<new_db_name>","<new_schema_password>"`
	`create database <new_db_name>`
	`exec sp_dboption '<new_db_name>', 'select into/bulkcopy/pllsort', true`
	`use <new_db_name>`
	`sp_adduser <new_db_name>`

DB2	`CREATE DATABASE <new_db_name>AUTOMATIC STORAGE YES USINGCODESET UTF-8 TERRITORY US COLLATE USING SYSTEM PAGESIZE 32 K;`
	`CONNECT TO <new_db_name>;`
	`CREATE SCHEMA <new_db_name>AUTHORIZATION <new_db_name>;`
	`GRANT CREATEIN ON SCHEMA <new_db_name>TO`

Database	Mobiliser Platform Schema Changes
	SY365_OBJOWNER; GRANT SY365_OBJOWNER TO USER <new_db_name>;
Oracle	DROP USER <new_db_name>CASCADE; CREATE USER <new_db_name> IDENTIFIED BY <new_db_password> GRANT SY365_OBJOWNER TO <new_db_name>; GRANT CREATE SESSION TO <new_db_name>; GRANT CONNECT TO <new_db_name>; ALTER USER <new_db_name>QUOTA UNLIMITED ON USERS; ALTER USER <new_db_name> DEFAULT ROLE ALL;

Note: If any the database credentials are changed, ensure to make note of all of the changes. If any of the database credentials are changed, ensure that the corresponding Mobiliser properties files are changed as well. Any access Mobiliser Platform needs for the database, such as configuration and properties files, must utilize the new credentials. For details, see *MOB2DB* on page 9.

Mobiliser Platform User

The Mobiliser Platform user is an internal application user that connects Web portals to the Mobiliser Platform and other internal tasks, for example, jobs that run inside the platform.

The password for this user is set during the initial setup of Mobiliser Platform, so there is no default password provided.

To change the password, follow the same steps required to set up the initial password. For details, see *Update the Default Configuration (mob-aps-1)* section in the *Sybase Mobiliser Platform Installation* guide.

Mobiliser Platform User

Configuration

Perform the post installation configuration tasks for the Mobiliser Platform components that compose your system.

Logging

Logging for the Mobiliser Platform Container is configured in:

```
/opt/sybase/money/conf/cfgbackup/org.ops4j.pax.logging.
properties
```

It is a standard log4j configuration file. For details on configuration, refer to http://logging.apache.org/log4j/1.2/manual.html.

Mobiliser Platform has its own log appender, which has two changes to the default daily rolling file appender:

- **Context name** – is the last part of the URL when a service is called and is added to the name of the log file
- **sessionId** – is part of each request and is included in each line of the log file that deals with handling the corresponding request

To enable request and response tracing, update the value in:

```
log4j.logger.com.sybase365.mobiliser.framework.service.jsonau
dit.JsonAuditManager.log=TRACE, JSON
```

The log appender only records events related to the Mobiliser Platform container if set to false:

```
log4j.additivity.com.sybase365.mobiliser.framework.service.
jsonaudit.JsonAuditManager.log=false
```

If set to true, the log appender also records external protocol events that may or may not have anything to do with the Mobiliser Platform container, such as SOAP, XML or JavaScript object notation (JSON) in addition to the other information. A setting of true creates larger log files and makes it more difficult to diagnose an issue.

Logging for the Mobiliser Platform Proxy is configured in:

```
/opt/sybase/proxy/conf/cfgbackup/org.ops4j.pax.logging.
properties
```

Follow the logging properties as described for the Mobiliser Platform container logging file:

```
/opt/sybase/money/conf/cfgbackup/org.ops4j.pax.logging.
properties
```

Enabling Strong Encryption in Java Development Kit

A Java Development Kit (JDK) installation, by default, supports only AES encryption with 128-bit key length, which is considered to be insecure.

1. To enable strong cryptography on your Java virtual machine (JVM), download the Java Cryptography Extension (JCE) unlimited strength jurisdiction policy file from your JDK vendor.

   For Oracle and IBM JDKs, two files are provided:

   - `local_policy.jar`
   - `US_export_policy.jar`
2. Replace these files in your JDK installation directory at:

   `/jre/lib/security`
3. Refer to the accompanying installation instructions for JVM specific hints.

Installing Bouncycastle Java Cryptography Extension

(Optional) You can create a secure channel with customer's mobile phone number and sign tokens on the Secure Element (SE). On-Device Charging (ODC) uses the following crypto algorithms that are implemented by 99% of SE:

- **Encryption:** DESede/CBC/PKCS5Padding with 128-bit keys for ODC, used for opening the secure channel with the SE.
- **Hashing:** DESede/Mac64/ISO7816-4Padding with 192-bit keys, used by ODC for signing tokens on the SE.

The bouncycastle package implements those crypto algorithms for the ODC server and must be installed by downloading the latest version from the bouncycastle Web site.

1. Go to:

   http://www.bouncycastle.org/latest_releases.html.
2. Expand the contents into:

   `$JAVA_HOME/jre/lib/ext`
3. Add the new security provider to the `$JAVA_HOME/lib/security/java.security` file:

   ```
   security.providerXX =
   org.bouncycastle.jce.provider.BouncyCastleProvider
   ```

   where `XX` is the next index following the last provider index specified in the file.

Encryption of Configuration Files

All configuration files in the `/opt/sybase/money/conf/cfgbackup` folder support encrypted configuration values. The master key for encryption of these values is stored in the `/opt/sybase/money/conf/system.properties` file:

- `com.sybase365.arf.container.system.decryptionkey=<PASSWORD>`

- `com.sybase365.arf.container.system.decryptionkeylength=<128|256>`

The 256-bit key length works only if the JVM's encryption policy files are replaced. Any configuration value in the property files at `./conf/cfgbackup` can be encrypted. The decryption of these values happens transparently to the Mobiliser Platform application, using the key configured in `./conf/system.properties`. Inside the Mobiliser Platform container are encrypted values that are visible in clear text, which includes the Web console. To indicate that a value is encrypted, it must be prefixed with '`{enc}`' (without quotes). An encrypted entry must look like:

```
<KEY>={enc}<ENCRYPTED-VALUE>
```

If you want to disable encryption support in a single configuration file explicitly, simply add this key/value pair into that particular property file:

```
com.sybase365.arf.container.system.configadmin.
decryptproperties=false
```

The AES/CBC/PKCS5Padding encryption is used. The encrypted value is expected to be base64 encoded, the first 16 bytes are interpreted as the initialization vector (IV). The encryption key is derived from the password using PBKDF2HmacWithSHA1 hashing with the static salt {97,101,105,111,117,85,79,73,69} and 65536 iterations. The Mobiliser Platform container includes an executable JAR in the `./tools` folder to encrypt configuration values according to this specification.

Simply run:

```
./tools> java -jar com.sybase365.mobiliser.vanilla.cli-tools-
5.1.0.RELEASE-CLIEncrypterClient.jar <KEY> <TEXT>
[<KEYLENGTH>]
```

Note: In the Mobiliser Platform 5.1 installations, the encryption tool requires the installation of X Windows in the system environment to execute properly; however, for Mobiliser Platform 5.1 SP01 and later installations, this tool can be run without X Windows capability. The <KEY> must match the configured key from `./conf/system.properties`, <KEYLENGTH> is optional and defaults to 128 bits - 256 only work if you've updated your Java encryption policy file.

If you use the encryption tool to create hash values for Preferences, make note of the <KEY> value used to create the hash because it must be used in future configuration of the context.xml (JDNI Entry) specified in *Encryption in Preferences*.

Encryption in Preferences

Preferences configuration values can be stored encrypted in the MOB_PREFERENCES table. Encrypted preferences values must be prefixed with the used encryption algorithm, such as:

- `{AES-128-PBKDF2}<ENCRYPTED-VALUE>`
- `{AES-256-PBKDF2}<ENCRYPTED-VALUE>`

Decryption happens transparently to the using application; however, the developer using a particular preferences node must enable encryption-support for this node explicitly. Hence, unlike configuration property file encryption, this only works if the developer has set it up like that.

For the Mobiliser Platform container, the en/decryption key is configured in:

```
./conf/cfgbackup/com.sybase365.mobiliser.util.prefs.encryption.
aes.properties

   preferencesEncryptionKey=<KEY>
```

For applications using remote access to preferences, the en/decryption is configured through one of these (descending priority):

- **System property:** `-Dcom.sybase365.mobiliser.money.prefs.secret=<KEY>`
- **JNDI entry:** `<Environment description="Preferences key" name="prefs/secret" type="java.lang.String" value="<KEY>" />` (usually configured in <TOMCAT_HOME>/conf/server.xml)
- **Property file on class path:** `sybase-preferences.properties`

  with line: `encryption-secret=<KEY>`

The AES/CBC/PKCS5Padding encryption is used. The encrypted value is expected to be base64 encoded, the first 16 bytes are interpreted as the IV. The encryption key is derived from the password using PBKDF2HmacWithSHA1 hashing with the static salt {97,101,105,111,117,85,79,73,69} and 65536 iterations. The Mobiliser Platform container includes an executable JAR in the `./tools` folder to encrypt configuration values according to this specification.

Simply run:

```
./tools> java -jar com.sybase365.mobiliser.vanilla.cli-tools-
5.1.0.RELEASE-CLIEncrypterClient.jar <KEY> <TEXT>
[<KEYLENGTH>]
```

Note: In Mobiliser Platform 5.1 installations, the encryption tool requires the installation of X Windows in the system environment in order to execute properly; however, for Mobiliser Platform 5.1 SP01 and later installations, this tool can be run without X Windows capability. The <KEY> must match the configured key from one of the configuration places mentioned above, <KEYLENGTH> is optional and defaults to 128 bits - 256 only works if you've updated your Java encryption policy file.

Alternatively, once your system is up and running you can also log into the Operations Dashboard to change the preferences through the portal. Remember to use the consistent encryption key there as well.

System Properties for Configuration

The configuration file values as well as preference configuration values can be expressed in terms of parameters by system property placeholders. Using system properties is an easy way of making a configuration specific to an individual node. You can use the identical set of configuration files and shared preferences configuration across all nodes, but then provide individual configuration where necessary by setting system properties differently on each node.

Use the `${...}` syntax when using a system property in a configuration file or in preferences. For example, `key=${SYSPROP}` resolves the `key` to the value of the system property `SYSPROP`. You can leave the syntax empty if the system property is not provided. You can embed the `${...}` syntax in other content, for example, `key=conf${SYSPROP}` resolves to `key=configuration`.

You can provide system properties by adding a key-value pair to the conf/system.properties file, or by using the regular `-Dkey=value` syntax by adding them to the setenv.sh/bat file.

Performance Considerations

The standalone Mobiliser Platform container includes a default configuration that is appropriate for workloads that do not require high transaction rates. A container that is running with the default settings is generally limited to development and proof of concept scenarios. By making a few minor adjustments, you can scale the container to your available hardware resources, providing high transactional throughput with low latency tolerances.

The settings detailed below are those that have the highest impact to performance. Recommended values are provided in units as a factor of the number of available CPU cores on the host that is running the Mobiliser Platform container. Most scenarios require the same number of CPU cores on the host that is running the database server as the host or hosts that are running the Mobiliser Platform container.

Jetty Thread Pool

The Jetty thread pool settings provide the main worker threads that handle incoming connections before handing operations off to other subsystems. The default values are generally sufficient for most workloads. You can make minor efficiency gains by setting the minimum number of threads to the same value as the maximum number of threads. As long as there are no operating system constraints, running more Jetty threads than recommended does not negatively impact performance.

Configuration file: `/opt/sybase/money/conf/jetty.xml`

Requires container restart to take effect: Yes

Setting Name	Default Value	Modified Value
minThreads	8	16 * CPU cores
maxThreads	256	16 * CPU cores

Database Connection Pool

Depending on the transaction being performed, the Mobiliser Platform container requires multiple calls to the database to complete the transaction. A large number of connections are required for high volumes of concurrent transactions. As with the Jetty thread pool, you can make efficiency gains by setting the minimum number of connections to the same value as the maximum number of connections. The total number of database connections is the product of the number of partitions and the connections per partition.

If there are no operating system or database restrictions, running a higher number of database connections than the recommended value does not negatively impact performance. The same is also true for the number of partitions.

Configuration file:
`/opt/sybase/money/conf/cfgbackup/com.sybase365.mobiliser.framework.persistence.jdbc.bonecp.pool.properties`

Requires container restart to take effect: Yes

Setting Name	Default Value	Modified Value
maxConnectionsPerPartition	10	16 * CPU cores
minConnectionsPerPartition	1	16 * CPU cores
partitionCount	2	N/A (default is recommended)

Java Memory Settings

You can control the amount of memory allocated to the Mobiliser Platform container by specifying the heap size for the Java command line process. The heap size represents only part of the memory used by Java. Depending on the operating system, the actual amount of memory used by Java can be two or three times the amount that is allocated to the heap.

You can make minor efficiency gains by setting the minimum amount allocated to the same value as the maximum value allocated. However, due to the garbage collection algorithm, performance can be negatively impacted by allocating more than 16GB to the heap. We strongly recommend that you do not exceed this value, regardless of the number of available CPU cores.

Configuration file: `${mobiliser_home}/bin/setenv.sh`

Requires container restart to take effect: Yes

Setting Name	Default Value	Modified Value
-Xms	256M	512M * CPU cores (not to exceed 16GB)
-Xmx	1G	512M * CPU cores (not to exceed 16GB)
-XX:PermSize	128M	256M
-XX:MaxPermSize	256M	N/A (default is recommended)

Log Levels

By default, the Mobiliser Platform container ships with relatively safe log levels configured. Relatively safe log levels means that the default is set into a capacity where there is not a lot of detail recorded in the logs. The amount of detail in log files can have a negative performance impact on the Mobiliser installation. The default setting provides the least amount of detail in the log, providing the best performance that the systems specifications can provide. More than one log file exists, but the log levels mentioned are all set in the properties file:

`org.ops4j.pax.logging.properties`

However, you may need to increase log levels during troubleshooting. In a high-throughput scenario, any log level that is set higher than “WARN” (especially “DEBUG” and “TRACE”) consumes potential resources.

Configuration file:
`/opt/sybase/money/conf/cfgbackup/org.ops4j.pax.logging.properties`

Requires container restart to take effect: No

Setting Name	Default Value	Modified Value
log4j.logger	Varies	WARN

Event Handling

Event generation and processing is an integral part of the underlying Mobiliser Platform container framework. The default values for the number of event-handling threads can create a situation where the events generated during a transaction can be processed after the transaction. This might reduce database load to a minor degree, but it can also require regenerative event processing if a container restart becomes necessary. You can largely avoid this effect by using the built-in auto detect logic when sizing the event-handling thread pools. In a high-throughput scenario, using this logic generally leads to more consistent load patterns.

Before being processed, events are stored in virtual queue structures inside the Mobiliser Platform container. Review the configuration settings for these parameters to prevent repeated error messages due to the structures being undersized.

Configuration file:
`/opt/sybase/money/conf/cfgbackup/com.sybase365.mobiliser.framework.event.core.properties`

Requires container restart to take effect: Yes

Setting Name	Default Value	Modified Value
regeneration.batch.size	10	1024 * CPU cores
delayedq.capacity	1000	65536 * CPU cores
processq.capacity	1000	65536 * CPU cores
catchupq.capacity	1000	65536 * CPU cores

Configuration location: Mobiliser Platform preferences (multiple locations)

Requires container restart to take effect: No

Setting Name	Default Value	Modified Value
event.handler.maxActive	Varies	-1
event.handler.maxIdle	Varies	-1

Business Logic Configuration

All relevant configuration options, such as Preferences and ConfigAdmin, are provided in this section for Mobiliser Platform components.

The ConfigAdmin PID relates to the configuration file of the same name in:

```
/opt/sybase/money/conf/cfgbackup/
```

Framework

The Mobiliser Platform Framework configuration is done through the ConfigAdmin option. Individual configuration options are described in the following sections.

Gateway

HTTPService

The HTTPService configures the main servlet used to process all HTTP requests coming into the framework.

ConfigAdmin PID:
`com.sybase365.mobiliser.framework.gateway.httpservice`

The HTTP service configuration options map directly to the underlying spring class:

- http://static.springsource.org/spring/docs/3.1.x/javadoc-api/org/springframework/web/multipart/commons/CommonsFileUploadSupport.html
- http://static.springsource.org/spring/docs/3.1.x/javadoc-api/org/springframework/web/multipart/commons/CommonsMultipartResolver.html

The HTTP service has the following configuration options:

Table 7. HTTP Service Configuration Details

Key	Default	Description
servlet	/mobiliser	Defines the name of the Mobiliser Platform servlet. Runtime updates to this value are ignored.
mode	standard	Defines the mode for this Mobiliser Platform servlet. Available options are standard and proxy. Proxy mode is only used for the validating proxy container.
multipartResolverType	commons	Defines which handler to use for multipart servlet requests. Currently supports only commons for commons-fileupload. See Table 8. HTTP Service DoS Filter Configuration Details on page 30 for specific configuration options.

Key	Default	Description
dos_filter_enabled	false	Defines the switch for Jetty DoSFilter. See Table 8. HTTP Service DoS Filter Configuration Details on page 30 for specific configuration options.
qos_filter_enabled	false	Defines the switch for Jetty QoSFilter. See Table 9. HTTP Service QoSFilter Configuration Details on page 31 for specific configuration options.
gzip_filter_enabled	true	Defines the switch for Jetty GzipFilter. See Table 10. HTTP Service GzipFilter Configuration Details on page 32 for specific configuration options.
commonsMultipartResolver.defaultEncoding	ISO-8859-1	Sets the default character encoding to use for parsing requests, to be applied to headers of individual parts and to form fields. Default is ISO-8859-1, according to the Servlet spec.
commonsMultipartResolver.maxInMemorySize	10240	Sets the maximum allowed size (in bytes) before uploads are written to disk. Uploaded files are received past this amount, but they are not stored in memory.
commonsMultipartResolver.maxUploadSize	-1	Sets the maximum allowed size (in bytes) before uploads are refused. The default value of -1 indicates no limit.
commonsMultipartResolver.resolveLazily	false	Sets whether to resolve the multipart request lazily at the time of file or parameter access.

The configuration options map directly to the filter configuration options of the underlying Jetty DosFilter, with each option prefixed with DOS.

For more details, see:

http://wiki.eclipse.org/Jetty/Reference/DoSFilter

Table 8. HTTP Service DoS Filter Configuration Details

Key	Default	Description
dos.delayMs	100	Defines the delay imposed on all requests over the rate limit, before they are considered at all. • **-1** rejects request • **0** no delay

Key	Default	Description
dos.insertHeaders	true	Inserts the DoSFilter headers into the response.
dos.ipWhitelist	127.0.0.1	Indicates a comma-separated list of IP addresses that are not rate limited.
dos.maxIdleTrackerMs	30000	Defines the length of time (in milliseconds) to keep track of request rates for a connection before deciding that the user has gone away and discarding it.
dos.maxRequestMs	30000	Defines the length of time (in milliseconds) to allow the request to run.
dos.maxRequestsPerSec	25	Defines the maximum number of requests from a connection per second. Requests in excess of this are first delayed, then throttled.
dos.maxWaitMs	50	Defines the length of time (in milliseconds) to blocking wait for the throttle semaphore.
dos.remotePort	false	Tracks the rate by IP+port (effectively connection) if set to true and session tracking is not used. Defaults to false.
dos.throttleMs	30000	Defines the length of time (in milliseconds) to asynchronous wait for semaphore.
dos.throttledRequests	5	Defines the number of requests over the rate limit able to be considered at once.
dos.trackSessions	true	Tracks usage rate by session if a session exists. Defaults to true.

The configuration options map directly to the filter configuration options of the underlying Jetty QosFilter, with each option prefixed with QOS.

For more details, see:

http://wiki.eclipse.org/Jetty/Reference/QoSFilter

Table 9. HTTP Service QoSFilter Configuration Details

Key	Default	Description
qos.maxPriority	10	Defines the maximum valid priority that can be assigned to a request. A request with a high priority value is more important than a request with a low priority value.
qos.maxRequests	10	Defines the maximum number of requests to be serviced at a time.
qos.suspendMs	-1	Defines the length of time (in milliseconds) that the request is suspended if it is not accepted immediately. If not set, the container's default suspend period applies.
qos.waitMs	50	Defines the length of time (in milliseconds) to wait while trying to accept a new request. Used when the maxRequests limit is reached.

The configuration options map directly to the filter configuration options of the underlying Jetty GzipFilter, with each option prefixed with GZIP.

For more details, see:

http://wiki.eclipse.org/Jetty/Feature/GZIP_Compression

Table 10. HTTP Service GzipFilter Configuration Details

Key	Default	Description
gzip.bufferSize	8192	Defines the output buffer size.
gzip.deflateCompressionLevel	-1	Defines the compression level used for deflate compression. (0-9).
gzip.deflateNoWrap	true	Defines the noWrap setting for deflate compression.
gzip.excludeAgentPatterns		Indicates a comma-separated list of user agents to exclude from compression but accepts regular expression patterns for more complex matching.
gzip.excludePathPatterns		Indicates a comma-separated list of paths to exclude from compression but accepts regular expression patterns for more complex matching.
gzip.excludePaths		Indicates a comma-separated list of paths to exclude from compression. Does a String.startsWith(String) comparison to check if the path matches. If it does match -> no compression. To match subpaths use excludePathPatterns instead.
gzip.excludedAgents		Indicates a comma-separated list of user agents to exclude from compression. Does a String.contains(CharSequence) to check if the excluded agent occurs in the user-agent header. If it does -> no compression.
gzip.methods		Indicates a comma-separated list of HTTP methods to compress. If not set, only GET requests are compressed.
gzip.mimeTypes		Indicates a comma-separated list of mime types to compress.
gzip.minGzipSize	256	Compresses the content if content length is either unknown or greater than value indicated.
gzip.vary	Accept-Encoding, User-Agent	Sets to the value of the Vary header sent with responses that could be compressed.

JMX Authentication

Java Management Extension (JMX) authentication configures an external listener for JMX connections into the framework.

ConfigAdmin PID:
`com.sybase365.mobiliser.framework.gateway.security.authentication.jmx`

The JMX authentication has the following configuration options:

Table 11. JMX Authentication Configuration Details

Key	Default	Description
jmxPort	5366	Defines the port to listen on for external JMX connections.
objectName	connector:name= rmi	Defines the object name under which this connector is registered as an MBean with the server.
serviceUrl	service:jmx:rmi:// 127.0.0.1:5366/jndi/ rmi://127.0.0.1: 5366/jmxrmi	Defines the URL that clients use to connect. Ensure that this port matches the JMX port.
initialContextFactory	com.sun.jndi.rmi. registry. RegistryContext Factory	Indicates which context factory to use.
exportInitialContextFactory	true	Defines whether to export the used initial context factory to the OSGi registry.
requiredRole	JMX_ACCESS	Defines the privilege required for access.
ssl	false	Defines whether to use an SslRmiServerSocketFactory. If this is set to true, you must additionally configure the keystore and truststore through the standard system properties: javax.net.ssl.

Web Console Base

Apache Felix Web Console is the base Web console tool used by Mobiliser Platform and is the central management interface for the underlying OSGi framework.

ConfigAdmin PID:
`org.apache.felix.webconsole.internal.servlet.OsgiManager`

If you are also deploying com.sybase365.mobiliser.framework.gateway.security.authentication.webconsole, the following settings are not used:

- `allowed.ip.list`
- `realm`

Instead, the standard Mobiliser Platform `realm` is in effect and the authentication is based on how it is done elsewhere in Mobiliser Platform.

Verify the settings for configuration details of the authentication provider.

The base configuration has the following options:

Table 12. Web Console Authentication Configuration Details

Key	Default	Description
manager.root	/system/console	Defines the root access URL for the Web console.
default.render	bundles	Defines the default tab to render.
allowed.ip.list		Indicates a comma-separated list of IP addresses that are allowed to access the Web console. If empty, access from anywhere is allowed.
plugins		Indicates a comma-separated list of plug-ins to activate.
realm	AIMS Management Console	Defines the authentication realm.

Web Console Authentication

The Web console authentication configures the authentication restrictions for the Web console based on the standard Mobiliser Platform security.

ConfigAdmin PID:
`com.sybase365.mobiliser.framework.gateway.security.authentication.webconsole`

It has the following configuration options:

Table 13. Web Console Authentication Configuration Details

Key	Default	Description
requiredRole	WEBCONSOLE_ ACCESS	Defines the role (privilege) for which a user must possess to be allowed access to the Web console.
requiredPort		Indicates the required port through which access to the Web console must take place.
whitelistedIps		Indicates a comma-separated list of IP addresses that are allowed to access the Web console. If empty, access from anywhere is allowed.
path	/system/console	Defines the URL for accessing the Web console.

Standard Security Filters

The standard security filters configures the standard security filters for Mobiliser Platform.

ConfigAdmin PID:
`com.sybase365.mobiliser.framework.gateway.security.filters.standard`

It has the following configuration options:

Table 14. Standard Security Filter Configuration Details

Key	Default	Description
ehCacheBasedUserCache.location	file:$mobiliser.home/conf/userdetails-ehcache.xml	Defines the location of the Ehcache configuration for the user details cache.
osgiProviderManager.eraseCredentialsAfterAuthentication	true	Defines whether Spring security removes the credentials from the authentication object after successful authentication. If you need to upgrade password hashing algorithms, this must be set to false since the password is needed to update the hash.
matcherMode	standard	Sets the HTTP path expressions. When matching HTTP paths for security expressions, Spring security normally uses the request path built by `request.getServletPath() + request.getPathInfo()`. For some environments, the path should be built with `request.getContextPath() + request.getPathInfo()`. For this mode, set this configuration to 'context.'
baseUrl	/mobiliser	Sets the base URL for Mobiliser Platform. Security configurations picked up from the OSGi registry can be relative or absolute. Relative configurations do not begin with a slash (/). In this case, the base URL configured here is prepended to the configuration before the HTTP path expression is configured. This should match the servlet name configured in the PID `com.sybase365.mobiliser.framework.gateway.httpservice`.
realmName	MOBILISER	Defines the realm name for the unauthorized response header. If the server receives a request for an access-protected object, and the request is denied, the server responds with a "401" response code and a "WWW-Authenticate" header.
channel	any	Defines the channel of the default security configuration for the Mobiliser Platform servlet. To use that particular channel to access the servlet, you can use: • any (default) • https • http You can override the default by providing specific configurations elsewhere within the container.
roles	MOBILISER_ACCESS	Defines the roles of the default security configuration for the Mobiliser Platform servlet, which is a comma-separated list and uses an 'OR' expression. You can override the default by providing specific configurations elsewhere within the container.

Key	Default	Description
port_mapping_xxx		Defines the mapping between secure and insecure ports. If a channel is set to something other than "any," whether with the default or other specific configuration, Spring security must know the mapping between secure and insecure ports to properly send the client a "302" response with a "Location" header. You may have any number of these configurations to specify the mappings between these ports. If you are using non-standard ports in jetty.xml for your connectors, you'll need to configure the ports here.

Mobiliser Platform uses a UserDetailsCache class for obtaining the user details that are used during authentication and authorization. It is configured via a standard EhCache XML file. In the standard setup, the location is configured in:

```
$mobiliser.home/conf/userdetails-ehcache.xml
```

The default EhCache configuration for the UserDetailsCache looks like:

```
1<?xml version="1.0" encoding="utf-8"?>
2<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
3  xsi:noNamespaceSchemaLocation="http://ehcache.org/ehcache.
xsd">
4
5  <defaultCache maxElementsInMemory="50" eternal="false"
6    overflowToDisk="false" memoryStoreEvictionPolicy="LRU"
7    timeToIdleSeconds="600" timeToLiveSeconds="0" />
8
9  <cache name="userDetailsCache" maxElementsInMemory="100"
10    eternal="false" overflowToDisk="false" memoryStoreEvict
ionPolicy="LRU"
11    timeToIdleSeconds="5" timeToLiveSeconds="5" />
12
13</ehcache>
```

Exception Mapping

Exceptions thrown by Mobiliser Platform endpoints are automatically caught and mapped to an error code, which is set in the MobiliserResponse object. The base MobiliserServiceException has an implicit error code field that is used for this mapping. Other exceptions are mapped to a constant value or are configured in this file to that value. To avoid mapping an exception (allowing it to pass), add a mapping to the configuration file and map it to nothing.

ConfigAdmin PID:
`com.sybase365.mobiliser.framework.service.api`

It has the following configuration options, which can be updated at runtime:

```
1
2# Simple mapping from exception class to error code
3# you can list parent classes here and subclasses
4# will be mapped to the error code of the parent
5
6# you can also add entries and map no number which will
7# cause the exception not to be mapped, but propagated
8
9org.springframework.dao.DataAccessException=9935
10org.springframework.dao.DuplicateKeyException=9930
11org.springframework.dao.DataIntegrityViolationException=9931
12org.springframework.jdbc.BadSqlGrammarException=9932
13org.springframework.orm.ObjectRetrievalFailureException=9933
14org.springframework.transaction.TransactionException=9935
15
16org.springframework.security.access.AccessDeniedException=
```

OData Interface

This configuration defines the open data protocol (OData) interface details.

ConfigAdmin PID:
`com.sybase365.mobiliser.framework.gateway.odata`

It has the following configuration options:

Table 15. OData Interface Configuration Details

Key	Default	Description
mode	standard	Sets the OData interface mode either as standard or proxy. Proxy is used only when deploying into the validating proxy container.
ignore_context_ regex	^\/?(prefs| management)$	Indicates the regular expression to be ignored. Any context matching this regular expression is not added to the OData interface.
path_match_ctx		Creates a new OData context from any configuration option with the prefix path_match_xxx. The value assigned is a regular expression, which is matched against contexts found in the container. If they match, this endpoint is sorted into this OData context. **Example:** `path_match_custom=^(bank|club)$` Any context matching bank or club is sorted into the OData context called custom. You can have any number of these mappings.
path_ privileges_ctx		Indicates a comma-separated list of privileges, which are required to access this OData context for any endpoints sorted into the context `ctx`. **Example:** `path_privileges_ctx=MY_ SPECIAL_PRIVILEGE`

Key	Default	Description
path_ports_ctx		Indicates a comma-separated list of ports, which must be used to access this OData context for any endpoints sorted into the context `ctx`. **Example:** `path_privileges_ctx=7070,8080`
default_context	odata	Sorts the endpoint into the default OData context if no matching path_match_xxx configuration is found.

TCP Interface

This configuration defines the TCP interface to Mobiliser Platform.

ConfigAdmin PID:
`com.sybase365.mobiliser.framework.gateway.tcp`

It has the following configuration options:

Table 16. TCP Interface Configuration Details

Key	Default	Description
port	8088	Defines the port to use for the TCP socket.
replyTimeout	10000	Defines the time (in milliseconds) for which the gateway waits for a reply.
useNio	false	Indicates whether or not the connection uses NIO.
applySequence	false	Sequences messages when using NIO. If attribute is set to true, correlationId and sequenceNumber headers are added to messages received.
soTimeout	0	Defaults to 0 (infinity), except for server connection factories with single-use="true". In that case, it defaults to the default reply timeout.
soSendBufferSize	0	See java.net.Socket.setSendBufferSize() located: http://docs.oracle.com/javase/6/docs/api/java/net/Socket.html#setSendBufferSize%28int%29
soReceiveBufferSize	0	See java.net.Socket.setReceiveBufferSize() located: http://docs.oracle.com/javase/6/docs/api/java/net/Socket.html#setReceiveBufferSize%28int%29
soTcpNoDelay	false	See java.net.Socket.setTcpNoDelay() located: http://docs.oracle.com/javase/6/docs/api/java/net/Socket.html#setTcpNoDelay%28boolean%29
lookupHost	false	Specifies whether reverse lookups are done on IP addresses to convert to host names for use in message headers. If set to false, the IP address is used instead of the host name.

Key	Default	Description
poolSize	false	This attribute is deprecated. For backward compatibility, it sets the backlog but users should use backlog to specify the connection backlog in server factories.
singleUse	false	Specifies whether a connection can be used for multiple messages. If set to true, a new connection is used for each message.
localAddress		Specifies an IP address for the interface to which the socket is bound on a multi-homed system.
taskSchedulerPoolSize	10	Defines the size of the task scheduler used by Spring Integration.
maxMessageSize	10240	Defines the maximum size allowed for a message - bridges TCP and Spring-WS.
endpointPathFilter	/smartphone, /spmmbanking	Indicates a comma-separated list of context paths that are not considered for use with the TCP interface.
ip_xxx		Specifies any number of options with the prefix ip_ followed by an IP address matcher. This should be mapped to a valid user in the system. The call is authenticated to this user. Access to the port from a particular host must be handled outside the TCP interface since it does no other checking that considering a host as being pre-authenticated with a particular user, such as: `ip_ 10.0.0.0/8=mobiliser`

The configuration options map directly to the IP endpoint configuration in Spring integration:

http://static.springsource.org/spring-integration/reference/htmlsingle/#ip-endpoint-reference

Hibernate

This configuration defines the properties for the Hibernate Session Factory. Runtime updates to these values are ignored.

ConfigAdmin PID:
`com.sybase365.mobiliser.framework.persistence.hibernate.sessionfactory`

The options for Hibernate configuration come directly from the Hibernate configuration documentation:

http://docs.jboss.org/hibernate/orm/3.3/reference/en/html/session-configuration.html

Additionally, it has the following configuration options:

Table 17. Hibernate EhCache Configuration Details

Key	Default	Description
ehCacheConfiguration	$mobiliser.home/ conf/mob-ehcache.xml	Defines the location of the cache configuration file.

JDBC

Currently, bundles for both BoneCP and C3P0 have been configured. However, only one should be used; not both.

BoneCP

This configures a BoneCP data source and can be updated at runtime.

ConfigAdmin PID:
`com.sybase365.mobiliser.framework.persistence.jdbc.bonecp.pool`

The options for this file come directly from the BoneCP configuration documentation:

http://jolbox.com/bonecp/downloads/site/apidocs/index.html?com/jolbox/bonecp/BoneCPConfig.html

C3P0

This configures a C3P0 data source and can be updated at runtime.

ConfigAdmin PID:
`com.sybase365.mobiliser.framework.persistence.jdbc.c3p0.pool`

The options for this file come directly from the BoneCP configuration documentation:

www.mchange.com/projects/c3p0/

Messaging

All message configurations are done through preference values. Individual configuration options are described in the following sections.

Engine

QueueBasedFutureFactory

The QueueBasedFutureFactory is responsible for managing the hand-off of responses in synchronous channel implementations.

`/com/sybase365/mobiliser/util/messaging/channelmanager/engine/impl/QueueBasedFutureFactory/`

It has the following configuration options in the preferences:

Table 18. Queue Based Future Factory

Key	Default	Description
synchronousPollTimeout	60000	Defines the amount of time (in milliseconds) the queued thread waits for an answer before returning null. This value can be updated at runtime.
synchronousOfferTimeout	60000	Defines the amount of time (in milliseconds) the response thread waits while trying to place the response on the queue before giving up. This value can be updated at runtime.

PickupJmsMessages

The PickupJmsMessages is responsible for picking up messages from configured JMS queues and forwarding them to ChannelManager to be sent through a channel implementation. This links the JMS feature with the OSGi ChannelManager.

```
/com/sybase365/mobiliser/util/messaging/channelmanager/engine/
jmspickup/PickupJmsMessages/
```

It has the following configuration options in the preferences:

Table 19. PickupJmsMessages

Key	Default	Description
sessionCacheSize	1	Defines the number of sessions to cache. Runtime updates are ignored. The pickup class makes use of a CachingConnectionFactory[a].
maxConcurrentConsumers	1	Specifies the number of consumers to create. Runtime updates only affect newly registered queues after the update. Listener containers that are already created are note destroyed and recreated. The pickup class makes use of a DefaultMessageListenerContainer[b]
mode	standard	Defines the mode for the picking up messages. For backwards compatibility with older Brand Mobiliser and Mobile Wizard instances set this value to legacy.

[a] Javadoc http://static.springsource.org/spring/docs/3.0.x/javadoc-api/org/springframework/jms/connection/CachingConnectionFactory.html

[b] Javadoc http://static.springsource.org/spring/docs/3.0.x/javadoc-api/org/springframework/jms/listener/DefaultMessageListenerContainer.html

JmsReceiveCallback

The JmsReceiveCallback is responsible for forwarding incoming messages to a JMS queues with the name of the destination supplied by the receiving channel. This links the JMS feature with the OSGi ChannelManager.

```
/com/sybase365/mobiliser/util/messaging/channelmanager/engine/
jmsreceiver/JmsReceiveCallback/
```

It has the following configuration options in the preferences:

Table 20. JmsReceiveCallback

Key	Default	Description
sessionCacheSize	1	Defines the number of sessions to cache. Runtime updates are ignored. The receiver class makes use of a CachingConnectionFactory[c].
mode	standard	Defines the mode for forwarding incoming messages. For backwards compatibility with older Brand Mobiliser and Mobile Wizard instances set this value to legacy.

[c] Javadoc http://static.springsource.org/spring/docs/3.0.x/javadoc-api/org/springframework/jms/connection/CachingConnectionFactory.html

ActiveMQConnectionConfiguration

The ActiveMQConnectionConfiguration is used to configure a JMS connection factory that is used by the OSGi ChannelManager components mentioned previously.

```
/com/sybase365/mobiliser/util/messaging/channelmanager/engine/
jmsconnection/ActiveMQConnectionConfiguration/
```

It has the following configuration options in the preferences:

Table 21. ActiveMQConnectionConfiguration

Key	Default	Description
brokerURL		Defines the URL of the JMS broker to use for the connection.
startBroker	false	Indicates whether the broker is to be started or not. If the value is set to true, the broker is started automatically.

Encryption

AesEncryptionStrategy

The AesEncryptionStrategy allows encrypting mail jobs with the AES-256 algorithm.

```
/com/sybase365/mobiliser/util/messaging/channelmanager/
encryption/aes/AesEncryptionStrategy/
```

It has the following configuration options in the preferences:

Table 22. AesEncryptionStrategy

Key	Default	Description
secret		Indicates the passphrase to use with the AES-256 algorithm for encrypting mail jobs. Alternatively, this value can be updated at runtime, but is not necessary.

TripleDesEncryptionStrategy

The AesEncryptionStrategy allows encrypting mail jobs with the 3-DES algorithm.

```
/com/sybase365/mobiliser/util/messaging/channelmanager/
encryption/tripledes/TripleDesEncryptionStrategy/
```

It has the following configuration options in the preferences:

Table 23. TripleDesEncryptionStrategy

Key	Default	Description
secret		Indicates the passphrase to use with the 3-DES algorithm for encrypting mail jobs. Alternatively, this value can be updated at runtime, but is not necessary.

Logic

MessagingLogicImpl

The MessagingLogicImpl is the core business logic class in the message gateway.

```
/com/sybase365/mobiliser/util/messaging/logic/impl/
MessagingLogicImpl/
```

It has the following configuration options in the preferences:

Table 24. MessagingLogicImpl

Key	Default	Description
defaultChannel		Allows clients of the message gateway to specify the name of the channel to use when sending a message. If nothing is specified, the default channel ID is used. This value can be updated at runtime.
disableEncryption	false	Allows message encryption to be overridden at the global level. Encrypted mail jobs are decrypted, but new jobs are written to the database in plain text. This can be useful when performing tests.

MessageDispatcher

The MessageDispatcher is used to poll the mail jobs table and dispatches messages to ChannelManager for sending.

```
/com/sybase365/mobiliser/util/messaging/logic/dispatcher/
MessageDispatcher/
```

It has the following configuration options in the preferences:

Table 25. MessageDispatcher

Key	Default	Description
interval	3000	Defines the interval between runs of the dispatcher (in milliseconds). This value can be updated at runtime.
batchSize	50	Defines the maximum number of mail jobs to select for processing during a run. This value can be updated at runtime.
maxInterval	60000	Defines the maximum interval if no jobs are found to dispatch. This value can be updated at runtime.
prefsBullyName	GW-MESSAGE	Defines the bully name to use to prevent duplicate processing of mail jobs. It can also be overridden through the system property: `com.sybase365.mobiliser.util.messaging.logic.dispatcher.MessageDispatcher.bullyService` Runtime updates of the preference value or the system property are ignored.

Template

CharsetUtilsImpl

The CharsetUtilsImpl is a utility class used by the template framework for choosing the best charset.

```
/com/sybase365/mobiliser/util/messaging/template/impl/charsets/
CharsetUtilsImpl/
```

It has the following configuration options in the preferences:

Table 26. CharsetUtilsImpl

Key	Default	Description
charsetList	us-ascii, iso8859-1, iso8859-15, utf-8	Indicates a list of charsets to be checked in ascending order (simplest to most complex). When choosing a charset, the engine bases its decision on this list until it finds a suitable charset for a string. This value can be updated at runtime.

BinaryContentTypeHandler

The BinaryContentTypeHandler is used to handle binary attachments in templates.

`/com/sybase365/mobiliser/util/messaging/template/impl/handlers/BinaryContentTypeHandler/`

It has the following configuration options in the preferences:

Table 27. BinaryContentTypeHandler

Key	Default	Description
defaultBinaryTypes	application/octet-stream, application/pdf, application/zip, application/x-gzip, image/gif, image/jpeg, image/png, image/tiff	Indicates a list of mime-types for which this content type handler is responsible. Runtime updates are ignored.

Channels

EmailChannel

The EmailChannel is used to send e-mail messages through simple mail transfer protocol (SMTP). It has the following configuration options in the preferences.

All values can be updated at runtime:

Table 28. EmailChannel

Key	Default	Description
channelId		Indicates the channel's ID.
mail.host	localhost	Defines the SMTP server host name.
mail.port	25	Defines the port to use when connecting to the SMTP server.
mail.username		Indicates whether the server authentication requires the user name.
mail.password		Indicates whether the server authentication requires the password.

Key	Default	Description
mail.protocol	smtp	Defines the protocol to use when speaking to the server. Options are SMTP or simple mail transfer protocol security (SMTPS).
mail.sign	false	Defines whether outgoing e-mail should be pretty good privacy (PGP) signed. All the remaining configuration values are only relevant if this value is set to true.
sign.keyId		Indicates the ID of the secret key to use for signing.
sign.hashAlgorithm	-1	Defines the hash algorithm to use when signing. If this value is not set, the default from the key itself is used. Otherwise, this option can be used to force a hash algorithm[d].
sign.passphrase		Indicates the passphrase for the secret key.
sign.secretRingResource		Indicates the resource for the secret keyring in Spring notation. For example, file:/home/mobiliser/.gnupg/secring.gpg.

[d] Javadoc http://www.bouncycastle.org/docs/pgdocs1.5on/org/bouncycastle/bcpg/HashAlgorithmTags.html

Note: The default channel for all outgoing messages is the `htmlchannel1`. Once an outgoing message is sent out by the Mobiliser Platform, the contents of the messages can be found at the following URL: https://mob-aps-1:8443/mobiliser/channelmgr/html, where you are asked for the Universal Mobiliser user credentials before viewing the outgoing messages. If you desire to use the `smtpchannel1` as the method for sending outgoing messages, you **MUST** deactivate `htmlchannel1` first. To deactivate it, you must set the "_active" preference entry to false via the Operations Dashboard in: `/com/sybase365/mobiliser/util/messaging/channelmanager/engine/impl/ChannelInstantiator/htmlchannel1`

HtmlChannel

The HtmlChannel is a useful test channel where outgoing messages are simply displayed in an HTML table under a specific URL. It has the following configuration options in the preferences.

All values can be updated at runtime:

Table 29. HtmlChannel

Key	Default	Description
channelId		Indicates the channel's ID.
maxSize	100	Defines the number of messages held in memory. Older messages are discarded.
urlSupplement	html	Appends the supplement to the channel manager URL making the HTML channel accessible.

HttpChannelEnd

The HttpChannelEnd is a useful test synchronous channel where messages can be submitted through HTTP.

It has the following configuration options in the preferences and all values can be updated at runtime:

Table 30. HttpChannelEnd

Key	Default	Description
channelId		Indicates the channel's ID.
urlSupplement	http	Appends the supplement to the channel manager URL making the HTTP channel accessible.
incomingChannelId		Indicates the channel ID used when this channel submits a message to ChannelManager. This is the destination.

JabberChannel

The JabberChannel is a useful test channel where messages are sent to a customer using extensible messaging and presence protocol (XMPP). The customer must first send a special message with slash (\) plus his mobile phone number (\+642222222) to the Mobiliser Platform Jabber user to register a mapping between his Jabber ID and a valid mobile phone number. The user specifies the short code to which the message should be sent by prefixing his text with the short code in brackets. Responses from the back end have the short code in brackets also preceding the text.

It has the following configuration options in the preferences and all values can be updated at runtime:

Table 31. JabberChannel

Key	Default	Description
channelId		Indicates the channel's ID.
urlSupplement	http	Appends the supplement to the channel manager URL making the HTTP channel accessible.
incomingChannelId		Indicates the channel ID used when this channel submits a message to ChannelManager. This is the destination.
xmppHost		Indicates the host name of the XMPP server.
xmppPort		Indicates the port to use for the XMPP server.
xmppUsername		Indicates the user name of the Mobiliser Platform Jabber user, for example, mobiliser.
xmppPassword		Indicates the password of the Mobiliser Platform Jabber user.
xmppServiceName		Indicates the host part of the Jabber user, for example, jabber.org.

Key	Default	Description
welcomeText	Welcome! Please enter your MSISDN with a leading \\(you can change it at any time)	Indicates the welcome message sent to a new Jabber ID.
instructionText	Please prefix your message with the shortcode in brackets, e.g. (234872) bal	Indicates the message instructing the Jabber user on how to communicate with the Jabber channel.

SmppChannel

The SmppChannel is used for sending/receiving SMS via SMPP. Many configuration values require knowledge of the SMPP:

http://www.nowsms.com/discus/messages/1/SMPP_v3_4_Issue1_2-24857.pdf

It has the following configuration options in the preferences and all values can be updated at runtime:

Table 32. SmppChannel

Key	Default	Description
channelId		Indicates the channel's ID.
urlSupplement	http	Appends the supplement to the channel manager URL making the HTTP channel accessible.
incomingChannelId		Indicates the channel ID used when this channel submits a message to ChannelManager. This is the destination.
sendAsDataSm	false	Sends SMS as a SUBMIT_SM. If set to true, then it sends SMS as DATA_SM.
registeredDelivery	0	Defines the value of the registered_delivery to use for outgoing SMS. See 5.2.17 of the SMPP Protocol Specification v3.4.
validityOffset		Defines the message validity period (in seconds) for an outgoing SMS. The default has no value, which delegates the validity offset responsibility to the defaulted defined in the short message service center (SMS-C).
smscTimeZone		Defines the time zone for SMS-C. If undefined, the SmppChannel assumes the same time zone as the JVM. When using validityOffset, the validity string must be computed using the time zone of the SMS-C.
ussdManagerNoReponseTimeout	300000	Defines the amount of time (in milliseconds) before a USSD session is regarded as timed out.
sourceNumberFormatter	const	Defines the source number format for outgoing messages. Options are include: • international • national

Key	Default	Description
		• const • numeric-international • short-international
destinationNumberFormatter	const	Defines the destination number format for outgoing messages. Options include: • international • national • const • numeric-international • short-international
srcTon	0	Defines the TON value to use for the source mobile phone number. This should match the configured formatter. See 5.2.5 of the SMPP Protocol Specification v3.4.
destTon	0	Defines the TON value to use for the destination mobile phone number. This should match the configured formatter. See 5.2.5 of the SMPP Protocol Specification v3.4.
srcNpi	0	Defines the NPI value to use for the source mobile phone number. This should match the configured formatter. See 5.2.6 of the SMPP Protocol Specification v3.4.
destNpi	0	Defines the NPI value to use for the destination mobile phone number. This should match the configured formatter. See 4.1 of the SMPP Spec.
bindType	transceiver	Defines the bind type to use when connecting to the SMS-C. Options include: • transceiver • transmitter • receiver • transmitterAndReceiver See 5.2.6 of the SMPP Protocol Specification v3.4. This setting affects all of the remaining options, listed with a leading underscore (_). Therefore, replace the underscore with transceiver, transmitter, or receiver. If you are using transmitterAndReceiver, you'll need to configure the following twice, once with each prefix.
_.host		Defines the host name of the SMPP server.
_.password		Defines the password of the user connecting to the SMPP server.
_.port		Defines the port number of the SMPP server.
_.systemId		Defines the system ID to use when binding to the SMPP server.

Key	Default	Description
_.systemType		Defines the system type to use when binding to the SMPP server.
_.enquireLinkTimer	60000	Defines the session timer (in milliseconds). If no activity is recorded, the underlying library sends an enquire link request to the server.
_.transactionTimer	10000	Defines the time (in milliseconds) to wait for the response to a sent request. For example, the length of time to wait for the SUBMIT_SM_RESP after sending a SUBMIT_SM.
_.initialReconnectDelay	1000	Defines the time (in milliseconds) to wait after starting the SmppChannel before attempting to connect to the server.
_.reconnectDelay	1000	Defines the time (in milliseconds) to wait before attempting to reconnect to the server after getting disconnected.
_.usingSSL		Indicates whether to use a secure sockets layer (SSL) connection when connecting to the SMPP server.
_.disableGsmAlphabet		Encodes the text in the global system for mobile communication (GSM) alphabet, if possible, and falls back to a multi-byte encoding. If you are using a language that normally cannot be encoded using the GSM alphabet, you can set this value to true to disable this check and always use a multi-byte encoding.

Note: The default channel for all outgoing messages is the `htmlchannel1`. Once an outgoing message is sent out by the Mobiliser Platform, the contents of the messages can be found at the following URL: https://mob-aps-1:8443/mobiliser/channelmgr/html, where you are asked for the Universal Mobiliser user credentials before viewing the outgoing messages. If you desire to use the `smtpchannel1` as the method for sending outgoing messages, you **MUST** deactivate `htmlchannel1` first. To deactivate it, you must set the "_active" preference entry to false via the Operations Dashboard in:
`/com/sybase365/mobiliser/util/messaging/channelmanager/engine/impl/ChannelInstantiator/htmlchannel1`

GCMChannel

The GCMChannel uses the Google Cloud Messaging (GCM) service to push key-value pairs to an Android application. It can only be used via sendPush messaging method.

It has the following configuration options in the preferences and all values can be updated at runtime:

Table 33. GCMChannel

Key	Default	Description
channelId		Indicates the channel's ID.
urlSupplement	gcm	Appends the supplement to the channel manager URL making the GCM accessible. This attribute is currently not used.
gcmUrl	GCM Service 1	Defines the URL of the GCM service. When the Mobiliser Platform must use a proxy server, configure the environment variables: • http(s).proxyHost • http(s).proxyPort

Audit

ConfigAdmin PID:
`com.sybase365.mobiliser.framework.service.audit`

Table 34. AuditDispatcher

Key	Default	Description
disabled	false	Disables the complete auditing subsystem.
allowCoreThreadTimeOut	true	Defines the configuration of the internal executor service.
corePoolSize	3	Defines the configuration of the internal executor service.
keepAliveSeconds	60	Defines the configuration of the internal executor service.
maxPoolSize	2147483647	Defines the configuration of the internal executor service.
queueCapacity	0	Defines the configuration of the internal executor service.

By default, use a synchronous queue and an unlimited number of threads, such as queueCapacity=0 and maxPoolSize. Alternatively, you can use a fixed number of threads and a large queue capacity so that tasks are placed on to a queue and then processed by the fixed number of threads:

```
allowCoreThreadTimeOut=true
corePoolSize=10
keepAliveSeconds=60
maxPoolSize=10
queueCapacity=2147483647
```

Note: If you have a large queue, do not set the corePoolSize to 1 since the executor service waits for the queue to fill before spawning new threads, which ends up causing no concurrency. If the queue size is too small and no threads are available, the audit subsystem starts dropping audits.

Database Audit Manager

The AuditDispatcher is responsible for dispatching a worker to process the queued audits into the database.

It has the following configuration options in the preferences:

`/com/sybase365/mobiliser/money/auditmgr/DatabaseAuditManager/`

Table 35. DBAuditDispatcher

Key	Default	Description
disabled	false	Disables audit manager.

`/com/sybase365/mobiliser/money/auditmgr/AuditDispatcher/`

Table 36. AuditDispatcher

Key	Default	Description
interval	3000	Defines the interval (in milliseconds) for running the dispatch worker to write entries to the database.
batchSize	50	Defines the maximum number of entries to write into the database as part of a batch.
interval	60000	Defines the maximum value for the interval between runs. If no entries are found, the interval between runs is increased up to this maximum value.
maxConcurrent	3	Defines the maximum concurrent workers to process the load. If the worker finds entries to write in the database and there are more than the batch size left after the run, the worker submits additional workers.
shutdownWait	5000	Defines the shutdown wait time (in milliseconds) for the executor dispatching the workers. If set to 0, the executor shuts down immediately losing any queued audits. Otherwise, the executor stops accepting new workers and waits for processing to complete or until the defined amount of time elapses before forcing a shutdown.

JSON Audit Manager

The JSON audit manager can be used to safely dump incoming request information into a log file. The JSON audit manager can be configured through the preferences or through ConfigAdmin. The configuration values are the same.

Preferences Node:
`/com/sybase365/mobiliser/framework/service/jsonaudit/JsonAuditManager/`

ConfigAdmin PID: `com.sybase365.mobiliser.framework.service.jsonaudit`

It has the following configuration options in the Preferences or ConfigAdmin:

Table 37. JsonAuditManager

Value	Key Description
regex	Any number of keys with value regex. These are regex values used to mask out fields which should not be logged in clear text.
bean	Any number of keys with value bean. These are beans which should not be logged at all. This can be used to completely skip certain request / response types.
field	Any number of keys with value field. Any field with this name is masked before logging.
path	Any number of keys with the value path. These are path expressions to mask values from being logged in clear text.

The keys are free text so an example best describes how to configure the audit manager:

```
 # mask any fields name pin or password
pin=field
password=field

# mask any request bean named GetPreferencesRequest
com.sybase365.mobiliser.util.contract.v5_0.prefs.GetPreferenc
esRequest=bean
 # mask any fields matching this regex
com\.sybase365\.mobiliser\.money\.contract\.v5_0\.customer\.s
ecurity\.(Login
  CustomerRequestType|SetCredentialRequest)/credential=regex

com\.sybase365\.mobiliser\.money\.contract\.v5_0\.transaction
\.Authentication
  Continue/Credential/(payer|payee)=regex
com\.sybase365\.mobiliser\.util\.contract\.v5_0\.messaging\.(
CreateAttachment
  Request|FindAttachmentsResponse|GetAllAttachmentsResponse)/
attachment(s)?/
  content=regex

# mask any fields matching this path
com.sybase365.mobiliser.util.contract.v5_0.messaging.Template
MessageRequest
  Type/message/parameters/value=path
```

Event Handler

All event handlers share a certain configuration. The preferences path is usually equivalent to the event or task implementation class name.

Table 38. General Event Preferences

Key	Default	Description
event.retry.delay	60	Defines the retry delay time (in seconds).

Key	Default	Description
event.retry.maxRetries	3	Defines the maximum number of retries.
event.expiry	0	Defines the time after which an event is regarded as expired and can no longer be handled.
event.handler. maxActive	4	Controls the maximum number of objects that can be allocated by the pool at any given time, such as checked out to clients or idle awaiting checkout. When non-positive, there is no limit to the number of objects that can be managed by the pool at one time. When maxActive is reached, the pool is said to be exhausted.
event.handler. maxIdle	2	Controls the maximum number of objects that can sit idle in the pool at any time. When negative, there is no limit to the number of objects that can be idle at one time.
internal.dao. callerId	-1	Defines the caller ID of the internal user. Used as the customer ID when creating or updating database records (ID_CUSTOMER_ CREATION, ID_CUSTOMER_LAST_UPDATE).
internal.service. username		Defines the user name used to authenticate the service call in case a task, job, or event handler needs to make a call to a Mobiliser Platform service.
internal.service. password		Defines the password used to authenticate the service call in case a task, job, or event handler needs to make a call to a Mobiliser Platform service.

Tasks

All tasks share a certain configuration. The preferences path is usually equivalent to the task implementation class name.

Table 39. General Task Preferences

Key	Default	Description
task.cronpattern	0 0/5 * ? * *	Defines the cron pattern on which the Task execution is scheduled.
internal.dao. callerId	-1	Defines the caller ID of the internal user. Used as the customer ID when creating or updating database records (ID_CUSTOMER_ CREATION, ID_CUSTOMER_LAST_UPDATE).
internal.service. username		Defines the user name used to authenticate the service call in case a task, job, or event handler needs to make a call to a Mobiliser Platform service.
internal.service. password		Defines the password used to authenticate the service call in case a task, job, or event handler needs to make a call to a Mobiliser Platform service.

Jobs

All Mobiliser Platform jobs share a certain configuration. The preferences path is usually equivalent to the job implementation class name.

Table 40. General Job Preferences

Key	Default	Description
internal.dao. callerId	-1	Defines the caller ID of the internal user. Used as the customer ID when creating or updating database records (ID_CUSTOMER_ CREATION, ID_CUSTOMER_LAST_UPDATE).
internal.service. username		Defines the user name used to authenticate the service call in case a task, job, or event handler needs to make a call to a Mobiliser Platform service.
internal.service. password		Defines the password used to authenticate the service call in case a task, job, or event handler needs to make a call to a Mobiliser Platform service.

The other configuration is read from the table MOB_JOBS, which controls the execution schedule as well as an additional parameter string that is handed over to the Job upon execution.

Miscellaneous Configuration

This section contains various configuration items that are configured via Preferences.

SecurityEndpoint

`com.sybase365.mobiliser.money.services.umgr.SecurityEndpoint`

The SecurityEndpoint sends out system generated passwords. In case a specific channel should be used for sending out the notifications containing the generated credential, the channel name can be configured via Preferences:

Table 41. SecurityEndpoint Preferences

Key	Default	Description
messageChannel.email		Defines the message channel to use for e-mails.
messageChannel.sms		Defines the message channel to use for SMS.

auth-handler sms-aoc

The class SmsAocAuthenticationHandler implements the IAuthenticationHandler interface and can be used for consumer authentication in line with a transaction processing. The handlers leverage a pre-configured “application” in Brand Mobiliser to start an interactive SMS session with the customer.

The Preferences path for this handler is:
`com.sybase365.mobiliser.money.businesslogic.authentication.handlers.smsaoc.PreferenceConfiguration`

Table 42. SmsAocAuthenticationHandler Preferences

Key	Default	Description
regex.yes	(ja)|(yes)|(oui)|(si)	Defines the regular expression to use to validate the positive user response.
brand.base		Indicates the configured rest service URL of Brand Mobiliser.
brand.client		Indicates the configured client context for Brand Mobiliser service.
brand.shortcode		Indicates the configured short code to use in Brand Mobiliser.
brand.keyword.payer	payeraoc	Defines the configured keyword in Brand Mobiliser to launch the application implementing the payer Advice of Charge authentication process.
brand.keyword.payee	payeeaoc	Defines the configured keyword in Brand Mobiliser to launch the application implementing the payee Advice of Charge authentication process.

One-Time Password Generation

A service in the one-time password (OTP) business logic exists to send out and validate non-persistent OTPs. The handling is different for standard (persistent) OTPs and the required configuration is done via preferences.

The preferences path to configure OTP generation is:

```
com.sybase365.mobiliser.money.businesslogic.customer.
configuration.CustomerOtpConfiguration
```

Table 43. OTP Preferences

Key	Default	Description
channel		Indicates the channel in channel manager to use to send out the message, fallback in case channel.email or channel.sms is not set.
channel.email		Indicates the channel in channel manager to use to send out the OTP via e-mail.
channel.sms		Indicates the channel in channel manager to use to send out the OTP via SMS.
tokenLength	6	Defines the length of the OTP (token) that is to be generated.
otpTypeAuthToken	100	Defines the authorization token for the OTP type in use ().
tokenTimeToleranceMinutes	2	Defines the time tolerance (+/-) when verifying if the token is valid (time stamp is part of generated token before hashing).
smsTokenTemplate		Defines the name of the template to use when sending out OTP tokens (non-persisted).

Payment Handler SecurityConfiguration

Sensitive payment instrument data, such as credit card and bank account numbers, are stored encrypted in the database. The AbstractCardPaymentHandler provides a function to decrypt the credit card number with a key that is provided in a key store. All relevant configurations are stored in preferences.

The Preferences path to configure the keystore and related parameters is:

```
com.sybase365.mobiliser.money.
businesslogic.payment.handlers.configuration.
SecurityConfiguration
```

Table 44. Payment Handler SecurityConfiguration Preferences

Key	Default	Description
key.store		Indicates the full path and name to the keystore (either on the file system or Java classpath).
key.store.type	JCEKS	Defines the type of the keystore.
key.store. password		Indicates the password of the keystore – usually identical to the key.password.
key.alias		Indicates the name of the key in the keystore to use for the decryption.
key.password		Indicates the password of the key in the keystore - usually identical to the key.store.password.

TransactionConfiguration

Most transaction related configuration is done in specific database tables. Preferences configuration is only in done in very few places of the transaction related business logic.

The preferences path is:

```
com.sybase365.mobiliser.money.businesslogic.transaction.
TransactionConfiguration
```

Table 45. TransactionConfiguration Preferences

Key	Default	Description
disableOpenTransactionsCheck	false	Disables the check for open transactions for the payer. Depending on the system and use case configuration, it has to be ensured that there is only one "open" transaction (error code = 0 && status code = 0) for the payer of a transaction (mostly to match an incoming, asynchronous authentication request that must be matched to the transaction).
alwaysCreate AuthorisationCode	false	Creates a transaction authorization code even if the authorization fails.
defaultExpirationMinutes	20,160 (14 days)	Defines the default authorization expiry set when use case does not have a specific configuration.

DemandForPayment

Demand for payment is sent by customers to request money. An invoice is created for this purpose, assigned to the requested. There is some configuration required to create a new invoice type, invoice configuration, and invoice that is used as default values for all demand for payment type invoices. This is done in preferences.

The preferences path to configure demand-for-payment is:

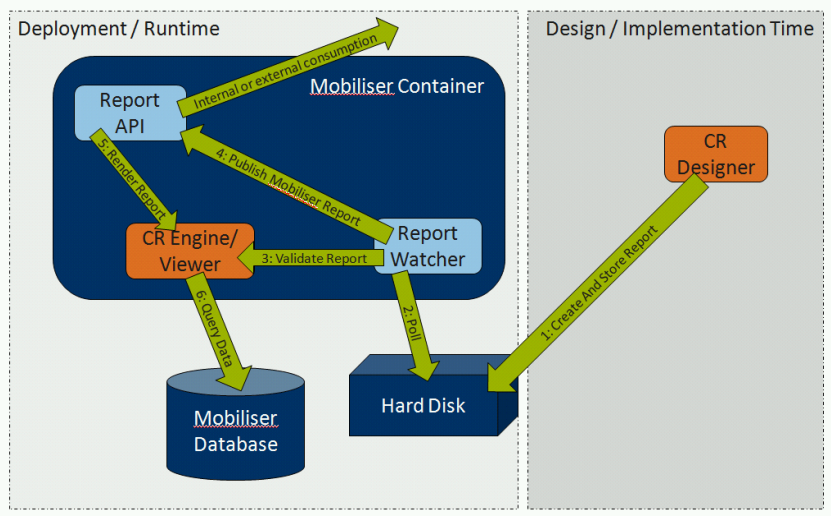
```
com.sybase365.mobiliser.money.businesslogic.
transaction.demandforpayment.impl.DemandForPaymentImpl
```

Table 46. DemandForPayment Preferences

Key	Default	Description
demandForPaymentUseCaseId		Defines the use case ID that is used when creating the invoice configuration and also when the applicable Payment Instruments are selected.
invoiceTypeHandlerTypeId		Defines the handler type ID of the bill payment handler that is to be used to execute demand for payment type Invoice payments.
invoiceTypeGroupId		Defines the default invoice type group ID that is to be used when creating new invoice types or when searching for existing invoice types.

Reporting Framework

SAP® Crystal Reports is used for designing and rendering the reports. Report design is an implementation task, and reports are simply added to the Mobiliser Platform installation by dropping them into the right place on the hard disk. Reports are discovered, validated through the Crystal Reports engine, and published through the reporting API.



Once a rendered report instance is requested, the Crystal Reports engine produces either a static output or a dynamic web-based view. Reports are rendered by pulling data from the Mobiliser Platform database.

Online Reports (Ad-hoc Reports)

Online reports can be requested and viewed by the Mobiliser Platform front-end. The parameters, which need to be supplied for the specific report to be generated, are retrieved from the RPT file by the report watcher when scanning for available reports. For each parameter the front-end renders a corresponding input field matching the type of the parameter, for example, date picker field for dates and drop down for “restricted” value sets. After the user provides the required parameters, a Web service request is sent returning the report view generated by the server, which is displayed by the front-end.

Asynchronous Reports

If the generation of the report takes a long time, you can request a report to be generated asynchronously. These reports are stored on the server and are available to be downloaded by the user who requested the report after it has been created.

You can also create asynchronous reports can by defining schedules for the reports using the cron expression format, such as once a day, once a month, or the last Friday of every Quarter. Defining such a schedule can be done in the front-end. A Mobiliser Platform job is created after providing the parameters required by the report, which triggers the creation of the report at the scheduled times.

Report Job

The report job uses the standard Mobiliser Platform job scheduler and service to register a report scheduler. The report scheduler is configurable through the database, starts generating reports through the reporting service per configured schedules, and supports pluggable workers to deliver the generated reports.

The cron job handler name is exposed with:

```
jobKey=com.sybase365.mobiliser.util.report.watcher.ReportJob
```

The format for tasks data is JSON, for example:

```
{"name":"TestReport1","locale":"en_US","format":"PDF", "repor
tParameters":[
    {"value":"v1","key":"p1","type":"java.lang.String"},
    {"value":1320702460403,"key":"p2","type":"java.util.Date"
}
   ]}
```

where the `"name"` parameter must match the report name already provisioned in the system. Also, the date parameter values are in milliseconds from epoch. Table 47. Common Report Parameters lists the JSON parameters, which are parsed by the job.

Table 47. Common Report Parameters

Name	Type	Description
name	java.lang.String	Defines the Report Name.
lastModifiedTime	java.lang.Long	(Optional) Defines the date of the last modification.
locale	java.lang.String	(Optional) Defines the location of the report.
format	java.lang.String	Defines the format for the report output: CSV, XLS, PDF.
owner	java.lang.Long	The report owner / customer id. If not set the generated report is available globally
reportParameters	reportParameters[]	A list of report parameters (value: key: type) see example

Report Store

The report store is used to store report templates as well as generated report instances. The live directory can hold any number of subdirectories. All reports in one subdirectory are logically grouped together and require the same user privilege to use these templates and generate reports.

The live subdirectories hold report templates that are available through the report services as well as through the front-end. The monitored directories and the required privileges for using these reports are configured through the ConfigAdmin file. Each directory can have an associated user interface privilege that decides the access rights to the execution of the report. The privileges are comma-separated entries from the report watcher properties.

For example:

```
#list of poll directories
pollDirectory=${mobiliser.home}/reports/live/distributor,
  ${mobiliser.home}/reports/live/mbanking
#poll directory privilege -
 matches privilege to poll directory
pollDirectoryPrivilege=MERCHANT_PORTAL_REPORTS,UI_CST_MBANKIN
G_REPORTS
```

The number of privilege entries should match the number of directory entries. In case the framework cannot process a report, or a report should be disabled manually, the report is moved into the archive directory. Asynchronous and batch reports are stored on the server indefinitely until they are they are altered in the database. The reports are stored in directories which, by convention, have the customerId of the customer requesting the report as folder name.

Cron Jobs

A Mobiliser Platform job is executed by a time trigger. Mobiliser Platform tasks can be used for multiple purposes such as house-keeping jobs or monitoring.

Mobiliser Platform jobs and tasks are very similar. However, there are a few differences:

- Job executions are managed centrally in the MOB_JOBS table. You can manage jobs in the Operations Dashboard portal.
- Jobs are only executed on a single server. Job management is synchronized via MOB_BULLY database table. Only one Mobiliser Platform container instance is responsible for executing all jobs. Fail-over handling is built in ensuring that each job only gets executed once for the given schedule.
- Jobs are more robust in terms of post-execution. In case of server outage, job history is maintained to track successful and failed job executions.

Mobiliser Platform provides a Cron Job Execution Task, which polls a database configuration table for job configurations and executes configured jobs (being Java classes) based on their schedule. The task ensures that each job does not run in parallel multiple times, cancels jobs that are unresponsive, and also can synchronize job execution across JVMs through a database based semaphore.

Cron Job Configurations

The cron job execution task retrieves jobs regularly from database tables according to the job handler names. The cron job task need to have cron job schedule and job handler names configured from the preference node:

```
com.sybase365.mobiliser.money.jobs.task.cronjob.handler.
CronjobTaskHandler
```

Task	Description
task.cronpattern	Defines the cron job schedule in cron expression format. The default expression is `0 0/1***?`, which is triggered every minute. Refer to the *Mobiliser Platform Framework Development* guide appendix for cron expression references.
task.jobHandlerNames	Defines the job handler names that correspond to the value in database table MOB_JOBS.STR_ HANDLER_NAME. The cron job tasks only pick up jobs that are configured to be handled by one of the names in this list. This can be useful when you want certain jobs to be executed by only a specific server or jobs being executed on all servers. Usually only one Mobiliser Platform instance is responsible for executing the jobs by getting a lock on individual handler names. Default expression: MOBILISER Multiple names should be separated by comma (,). For example, `MOBILISER1,MOBILISER2,MOBILISER3`

By configuring individual handler names, each server executes only the specific jobs. If you set up new handler names, make sure to also create new records in the MOB_BULLY table which is used to manage and maintain the lock.

In case you want to use this feature, it is best to configure the name of a system property as the value for the task.jobHandlerNames key, for example `${JOB_HANDLER_NAMES}`. This value is replaced by the system property with the same name. The system property can be set individually for each server by adding a `-D` parameter to the start of the JVM, for example `-DJOB_HANDLER_ NAMES=MOBILISER,MOBILISER2`.

Job Configuration

The jobs are configured in database table MOB_JOBS.

Table 48. Job Configuration - Mandatory Fields

Field	Description
ID_JOB	Indicates the unique integer to identify a job.
STR_HANDLER_NAME	Defines the handler name. **Note:** This handler name is to define which cron job task handles this job. It is different with the Mobiliser Platform task handler name described in events system.
STR_SCHEDULE	Defines job schedule in cron expression format. For more information, see Appendix B of the *Developer's Guide*.
ID_IMPLEMENTATION_TYPE	Defines the meaning of the URL_IMPLEMENTATION string. The value 2 means using full class name and the value 3 means using job bean name as filter string.
URL_IMPLEMENTATION	Defines the service filter to apply to find the proper implementation of IMobiliserCronJob in the OSGi service registry. Usually you'd just use the full class name with the package name where the job is implemented if ID_IMPLEMENTATION_TYPE=2. Otherwise use the job bean name if ID_IMPLEMENTATION_TYPE=3.
BOL_IS_ACTIVE	Indicates whether job is active or not. Set to "Y" if active, otherwise set to "N".

Table 49. Job Configuration - Optional Fields

Field	Description
DAT_LAST_EXECUTION	Indicates the date the job was last executed. Set to NULL initially.
INT_MAX_DELAY	Defines the max number of minutes a job is executed after the scheduled time (e.g. in case of system restart).
INT_MAX_DURATION	Defines the max duration in minutes. After this time the job is handled as failed.
STR_JOB	Defines the description of the job.
STR_PARAMETER	Defines the parameters that the job handler requires for job processing.

Field	Description
DAT_CREATION	Defines the date the job is created.
ID_CUSTOMER_CREATION	Defines the customer ID of who created the job.
DAT_LAST_UPDATE	Defines the date the job was last updated.
ID_CUSTOMER_LAST_UPDATE	Defines the customer ID of who last updated the job.

Bully Mediator Setting

The MOB_BULLY table is used to synchronize between multiple Mobiliser Platform instances for access to resources that must only be handled by a single instance, such as Job execution.

Make sure you have one bully entry for each jobHandlerName configured, such as MOBILISER, MOBILISER1, MOBILISER2. For example, for the default handler name MOBILISER, there should be an entry with the STR_SERVICE="MOBILISER" in table MOB_BULLY. If not, insert a new one with the value below:

Table 50. Bully Mediator Setting

STR_SERVICE	BOL_IS_ACTIVE	INT_LEASE_TERM
MOBILISER	Y	300

Security Fundamentals

Mobiliser Platform uses a set of security mechanisms to:

- Prevent unauthorized access
- Secure network configuration
- Block access to sensitive information

The following sections highlight the various mechanisms in place and provide the background information to understand the required configuration that must be done to harden the applications.

Prevent Unauthorized Access

All application level user accounts are managed by Mobiliser Platform in its central database. The relevant tables are:

- MOB_CUSTOMERS – stores information about the type of user and the status information.
- MOB_CUSTOMERS_IDENTIFICATIONS – stores unique identifications for the users, such as user names.
- MOB_CUSTOMERS_CREDENTIALS – stores hashed passwords of the users.

Other than for the initial setup, there is usually no direct interaction with the tables required.

Based on various criteria specific to the individual customer or the customer type, the users are equipped with privileges that allow them to make certain service calls into Mobiliser Platform.

Each service call is normally protected by an individual privilege. In addition, the service is bundled into a context. The context may require another privilege and defines the URL under which a service is exported.

In some situations technical users are used, for example, by the Web portal or by other internal components that invoke services.

Web Portal Access to Mobiliser Platform

The Web portals use two different system level users to gain access to Mobiliser Platform services.

The first user is defined along with the URL to the preferences service as a Java Naming and Directory Interface (JNDI) string resources (usually in the conf/context.xml in tomcat).

The configuration URL consists of multiple parts:

```
<scheme>://<user>:<password>@<host:port>/<path>?pollInterval=
<interval>&clientType=<clientType>&applicationIdentifier=<app
licationIdentifier>
```

Note: Make sure that you have the URL XML encoded when storing the URL in an XML file, for example use `&` to display the ampersand (&).

Table 51. Web Portal Access to Mobiliser Platform

Fragment	Description
scheme	Defines the scheme values: `prefs` and `prefss`. If prefss is set, the connection is done via HTTPS; otherwise, HTTP is used.
User	Defines the user that is used to access the preferences services. The default user is `prefsread`, which has the minimum set of roles configured to access the services.
password	Defines the password that must match the password of the configured user in Mobiliser Platform.
host:port	Defines the host name and the port that Mobiliser Platform is accessible.
path	Defines the URL path, usually mobiliser/rest/prefs, to the service.
interval	Defines the time interval (in milliseconds) that checks for new configuration with the server.
clientType	Indicates the protocol to use when calling the service. Default is "json".
applicationIdentifier	Identifies the configuration set being used. Mobiliser Platform preferences option manages multiple configuration trees. Each one is identified by an applicationIdentifier. Usually there are two being used in the standard configuration: • businesslayer – is for the back end (Mobiliser Platform container) • presentationlayer – is used by the Web portals Read and write access to the applicationIdentifiers can be granted individually.

The environment user (prefs2/config) is used by the Tomcat instance (Web_UI) to access the Mobiliser Platform Preferences, which is listening on a specific mob-aps-1 port. The user only has access to the services that allow read access to the preferences information.

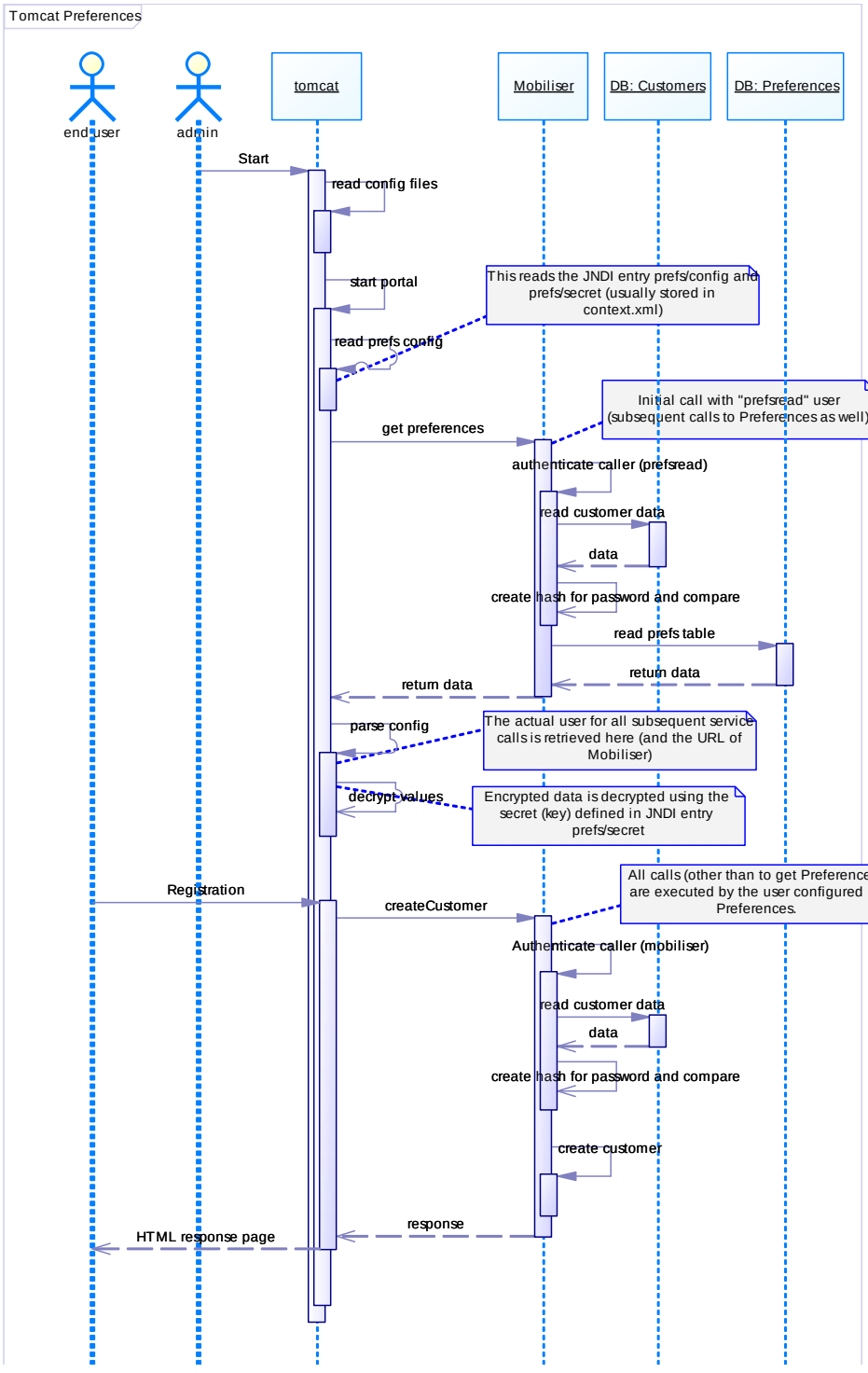
The configuration data from preferences contains all other application level configuration, including the user that is used for the subsequent service calls from the Portal to Mobiliser Platform.

Some data, such as passwords, are stored encrypted in the preferences. The key to decrypt the data is also stored as a JNDI string resource, with the name "prefs/secret". If you change this value you need to re-encrypt all the encrypted data stored in preferences (for the given applicationIdentifier). This can be done with the provided command line tool and SQL or by using the Operations Dashboard preferences functionality.

By default the Web portal uses the "mobiliser" user to make service calls to the back end. User name and password are configured in Preferences. The password which is stored encrypted in the Preferences must match the hashed password that is stored for the corresponding user (mobiliser) in the Mobiliser Platform database (MOB_CUSTOMERS_CREDENTIALS).

For security reasons, this password is not set in the standard configuration and must be set manually during the installation and setup process of Mobiliser Platform to the

MOB_CUSTOMERS_CREDENTIALS and to the MOB_PREFERENCES tables, once hashed and once in encrypted format.



Additional Information on Hashing Customer Credentials

Any customer (consumer, merchant, agent, or system user) credentials are stored hashed in MOB_CUSTOMER_CREDENTIALS. Mobiliser Platform supports using different hashing algorithms. The STR_CREDENTIAL is always prefixed with the hashing algorithm in curly brackets, for example, {<HASH-ALGORITH>}<HASHVALUE>.

Configuration of hashing algorithms is controlled through preferences. Update the following node:

```
com.sybase365.mobiliser.money.businesslogic.umgr.impl.
SmartPasswordEncoder
```

Table 52. Hashing Algorithm Preferences

Key	Description
Algorithms	Indicates a comma-separated list of supported hashing algorithms. The default list is: `SHA,SHA-256,SHA-512,SHA-512:1,SHA-512:10000,PBKDF2WithHmacSHA1:10000,BCRYPT:10,SSHA-512:10000,SPBKDF2WithHmacSHA1:10000`
encodeAlgorithm	Defines the algorithm to use for storing and encoding new credentials. The default is: `SSHA-512:10000`
defaultAlgorithm	Defines the algorithm to use for credential validation if the algorithm is not specified with the stored credential. The default is: `SHA`

You can change the default configurations within certain boundaries. You may only add new hashing algorithms when they are provided through JCE—that is, the hashing algorithms either come with your JDK or you've installed an extension like bouncycastle into your JDK. However, you can change the number of iterations/strength, which is the numeric value after the colon, to either increase the performance or security if required.

Warning! BCrypt is decreasing performance tremendously, so only use that if this is a strong security requirement.

Mobiliser Platform also supports an upgrade of the used hash algorithm—that is, each time a customer's credential gets checked, Mobiliser Platform validates if the used hashing algorithm is configured to be updated with the configured 'encodeAlgorithm'. Update the following node:

```
com.sybase365.mobiliser.money.businesslogic.umgr.impl.
SecurityLogic
```

Table 53. Upgrade of Used Hash Algorithms

Key	Description
hashUpgradePattern	Identifies the hash upgrade pattern, which is a Java regular expression (regex) pattern class. If set to <null>, no password upgrade is performed; otherwise any hashed password that matches this pattern is re-hashed using the current 'encodeAlgorithm'. By default, this value is not configured. The idea is to decide write a negated regex. For example, to transition all hashes to BCRYPT:10, use: `^(?:(?!\{BCRYPT:10\})).+$`

Additionally, you must reconfigure Spring Security to allow access to the plain text password for rehashing. Update the following node:

```
com.sybase365.mobiliser.framework.gateway.security.filters.
standard
```

Set the osgiProviderManager.eraseCredentialsAfterAuthentication key to FALSE. When this key is set to false, the authentication object returned from Spring Security does not have the credentials cleared, allowing rehashing of plain text password.

The actual value stored in STR_CREDENTIAL depends on the used hashing algorithm. All hash values are base64 encoded. For all algorithms, which does not use random salts, the customer ID is used as the salt value. Random salts are always 16 byte.

```
SHA: BASE64(HASH(<SALT>|<HASH>))
SSHA: BASE64(<SALT>HASH(<SALT><HASH>))
PBKDF2: BASE64(HASH(<SALT>,<HASH>))
SPBKDF2: BASE64(<SALT>HASH(<SALT>,<HASH>))
BCrypt: $2a$<ROUNDS#>$BASE64(<SALT><HASH>)
```

Mobiliser Platform comes with a Java executable to compute hash values:

```
./tools> java-jar com.sybase365.mobiliser.vanilla.cli-tools-
5.1.0.RELEASE-CLIPasswordEncoderClient.jar
```

Secure Network Communication

Mobiliser Platform components are spread across different hosts, which is explained in *Standard Deployment Model* on page 1. Communication between the applications must be encrypted. Otherwise, sensitive information exchanged between components, such as the Web Portal to Mobiliser Platform, could be intercepted.

Encryption relies on HTTPS, which means that the required certificates must be installed on all applications that provide an HTTP(S) port. If possible, officially signed certificates should be used. For testing / demo purposes you can also use self-signed certificates.

For internal developer deployments, HTTPS can be disabled.

Proxy Setup

Use a proxy in the DMZ that provides restricted access to the services provided by Mobiliser Platform Container. This can either by done using a standard reverse proxy or the Mobiliser Platform Validating Proxy.

Security Considerations

If any services of the Mobiliser Platform Core must be exposed to the public Internet, such as the consumption by Smartphone Mobiliser, it is essential that only a subset of the services offered by Mobiliser Platform be exposed on the Internet. The privilege and role based security concept of Mobiliser Platform only grants access to services for users on an as-needed basis, but there is no need to expose all of the services on the Internet.

Services in Mobiliser Platform are always attached to a "context" that defines the last section of the URL to address a specific service.

Expose Web Service Endpoints Securely

The default context for generic customer related services is http://localhost:8080/mobiliser/customer. However, the secure context, according to the Standard Mobiliser Platform Deployment diagram, is located at https://mob-aps-1:8443/mobiliser/customer.

The default context for services to be consumed by Smartphone Mobiliser Platform is http://localhost:8080/mobiliser/smartphone. However, the secure context is located at https://mob-aps-1:8445/mobiliser/smartphone.

In addition to HTTP, Mobiliser Platform supports various transport protocols. The JSON services are exposed under a slightly different URL. The JSON variants to the two examples mentioned above are:

- https://mob-aps-1:8443/mobiliser/rest/customer
- https://mob-aps-1:8445/mobiliser/rest/smartphone

So in most cases it is sufficient to expose the following URLs from the Mobiliser Platform Core:

- https://mob-aps-1:8445/mobiliser/smartphone
- https://mob-aps-1:8445/mobiliser/rest/smartphone
- https://mob-aps-1:8443/mobiliser/binary
- https://mob-aps-1:8443/mobiliser/rest/binary

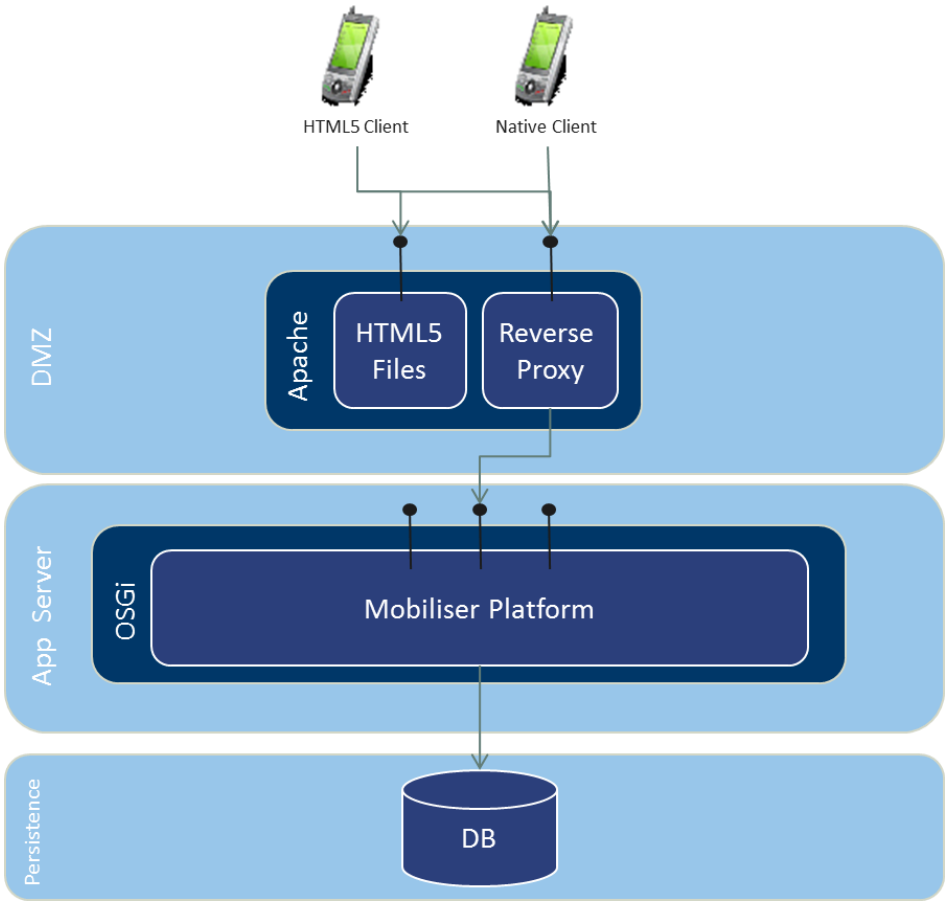
Some customized projects might use other or additional URLs.

Standard Reverse Proxy

Any reverse proxy, such as Apache HTTPD, can be used to accept incoming requests from the Internet in the DMZ and to forward them to the Mobiliser Platform Core running on the application server tier. See the *Sybase Mobiliser Platform Installation* guide for full installation and configuration instructions.

This example is provided for an Apache HTTPD server with proxy modules installed:

```
   <Proxy *>
        Order deny,allow
        Allow from all
   </Proxy>
ProxyPass /mobiliser/smartphone
http://localhost:8080/mobiliser/smartphone

ProxyPassReverse /mobiliser/smartphone
http://localhost:8080/mobiliser/smartphone

ProxyPass /mobiliser/rest/smartphone
http://localhost:8080/mobiliser/rest/smartphone

ProxyPassReverse /mobiliser/rest/smartphone
http://localhost:8080/mobiliser/rest/smartphone

ProxyPass /mobiliser/binary
http://localhost:8080/mobiliser/binary

ProxyPassReverse /mobiliser/binary
http://localhost:8080/mobiliser/binary

ProxyPass /mobiliser/rest/binary
http://localhost:8080/mobiliser/rest/binary

ProxyPassReverse /mobiliser/rest/binary
http://localhost:8080/mobiliser/rest/binary
```



In addition to shielding direct access to the Mobiliser Platform Core, the Apache server can be used to provide access to the HTML5 version of Smartphone Mobiliser, or any other HTML5 application. The same origin policy requires that the HTML files and the AJAX services be provided by the same server (host name + port). For more information about same origin policy, see http://en.wikipedia.org/wiki/Same_origin_policy.

The reverse proxy can also be used for the SSL termination. When you deploy Mobiliser Platform in the standard reverse proxy model, you must make changes to the configuration on the Apache Web UI container (located on the DMZ layer) and one of the database preferences (located in the persistence layer).

Default Web UI Accounts

When you install Mobiliser Platform, the installation process adds a set of predefined accounts. These accounts have special privileges required to administer the portal system configuration. Additionally, these accounts are used to create and manage user accounts, notifications and alerts, and merchants.

Note: After you log in using the predefined password, you are prompted to change it immediately before proceeding.

Account	Description
cstfull:secret	Full customer support privileges to the Administration Portal
usermgr:secret	Manage agent accounts
notifmgr:secret	Manage notifications and alerts
headquarter:secret	Create and manage merchants
opsmgr:secret	View and manage system configuration
sysmgr:secret	Monitor all functions of the Mobiliser Platform container

Changing Passwords for Default Web UI Accounts

1. Log into the internal Web UI portal with any of the default accounts.
2. Click **SELFCARE**.
3. Select **Change Password**.
4. Enter the old password for the logged in user.
5. Enter the new password.
6. Confirm the new password.
7. Click **Update**.

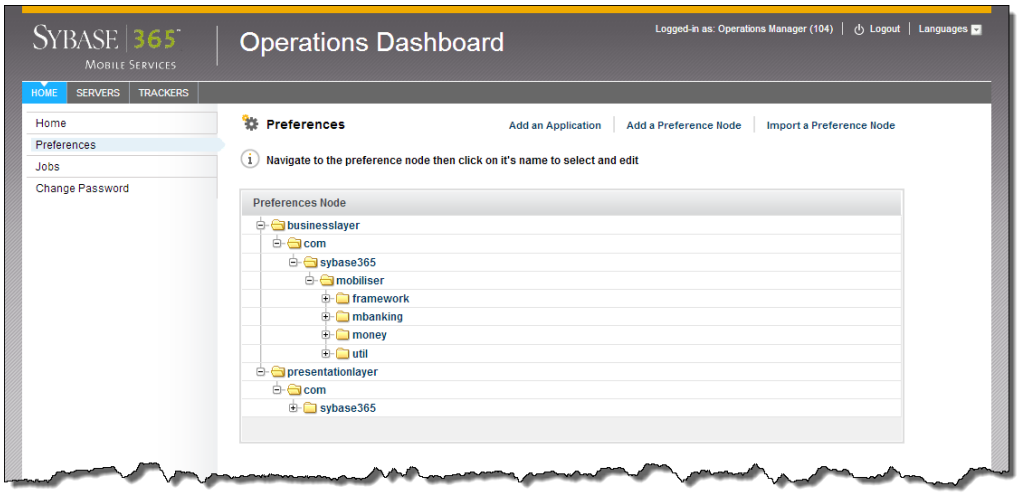
Operations Dashboard

The Operations Dashboard provides system administrators a high-level operational view into Mobiliser Platform. The dashboard aggregates the information to give you an overview on the application status, which aids in the operational support of the system. You can track individual statistics from your servers for monitoring the performance and general operational efficiency.

Preferences

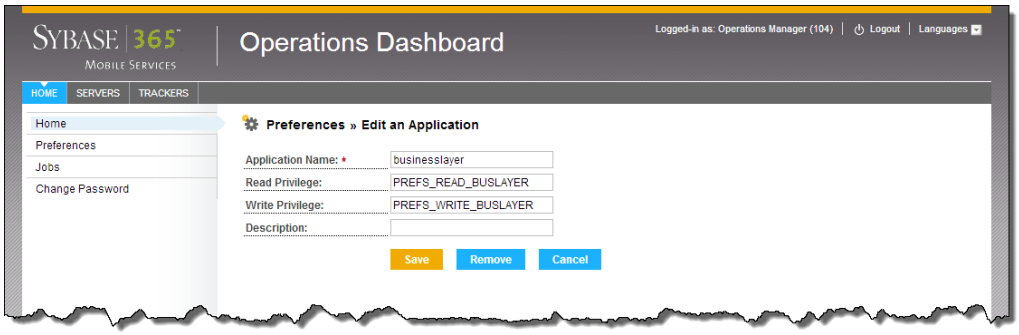
Preferences are the standard mechanism for application configuration in Mobiliser Platform. Use the Preferences option to manage operation-level configuration data such as timeouts, retries when communicating with other systems, and thread pool sizes. The standard Mobiliser Platform installation comes with two applications: businesslayer and presentationlayer.

You can add applications and preference nodes. Additionally, you can import the node information from an XML file, which contains the application name, path of the node, and preference keys and values. When you import the data it is added to the path of the node indicated in the XML file.



Applications

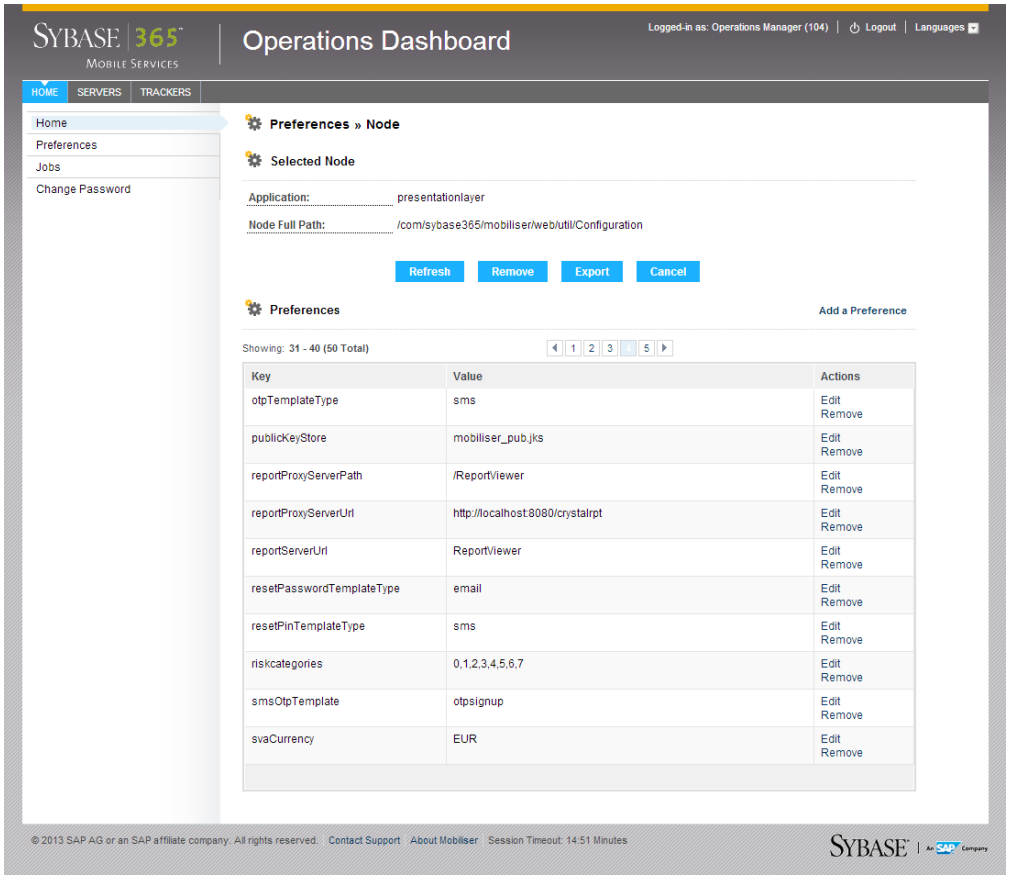
Preferences can be defined for multiple applications, each with a unique name and access rights. The application must have a unique name and may optionally have a description. You can define read or write privileges or both for the application. If read or write privileges are not defined, any user invoking Mobiliser Platform services can retrieve or set preference values. You can also edit or remove applications.



Node and System Preferences

Preference nodes store system preferences and configuration data. Each system preference contains a key-value pair associated with a preference path. You can add a preference node, which requires the application name and full node path. If the application does not exist, you must add the application before adding a preference node. You can remove preference nodes, but you cannot edit them.

Additionally, you can export the node information to XML, which contains the application name, path of the node, and preference keys and values.



Jobs

The Jobs option lets you schedule background jobs to run at certain times using cron expressions. For example, you can schedule a job to run at midnight to transfer commissions to the individual partners. You can also schedule a job to run every five minutes to generate new invoices. The cron job execution task retrieves jobs regularly from a database table according to the job handler name. The task makes sure that each job does not run in parallel multiple times, cancels jobs that are not responding, and synchronizes job execution across JVMs.

Field	Description
Handler	Defines which cron job task handles the job. The handler corresponds to a defined value in the database. The default handler is MOBILISER. You can enter multiple handlers separated by commas (MOBILISER, MOBILISER1, and so on).
Serviced By	Indicates the filter for the job implementation: Class or Bean.
Implementation	Defines the service filter to find the proper implementation in the service registry. • If the service filter is set to Class, then use the full class name. • If the service filter is set to Bean, then use the job bean name.
Schedule	Defines the job schedule in cron expression format. For example, 0 0/5 * ? * * * is a cron expression that is triggered every 5 minutes. The default expression is 0 0/1 * * * ?, which is triggered to run every minute.
Parameters	Defines the parameters that the job handler requires for job processing. Parameters can be any string that depends on the expected result of the job.
Job Name	Defines the description of the job.
Archive	Determines if the job is active or inactive.
Max Delay	Gives the maximum number of minutes a job is executed after the scheduled time. If max delay time is exceeded, the job is not started.
Max Duration	Gives the maximum duration in minutes. After this time the job is handled as failed. If max duration time is exceeded, such as running longer than indicated, the job is canceled and marked as failed.

See Also

- *Cron Expression Reference* on page 103

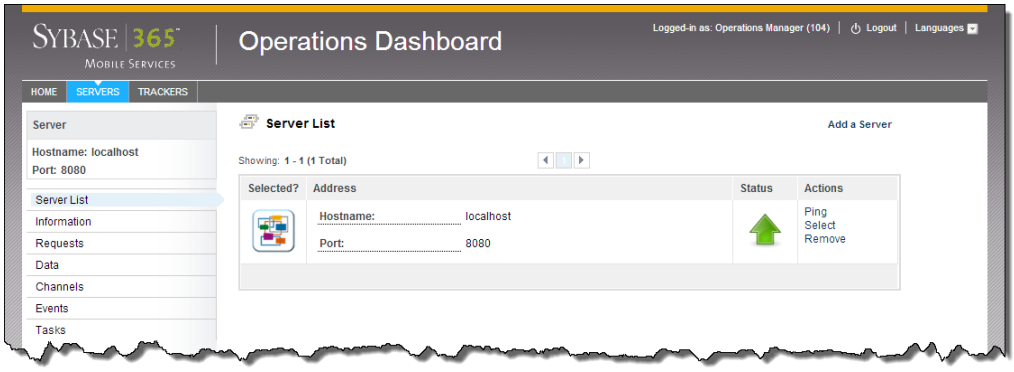
Servers

The Servers option displays a list of available Mobiliser Platform servers. You can select a server that is online to view information, requests, data, channels, events, or tasks.

Server List

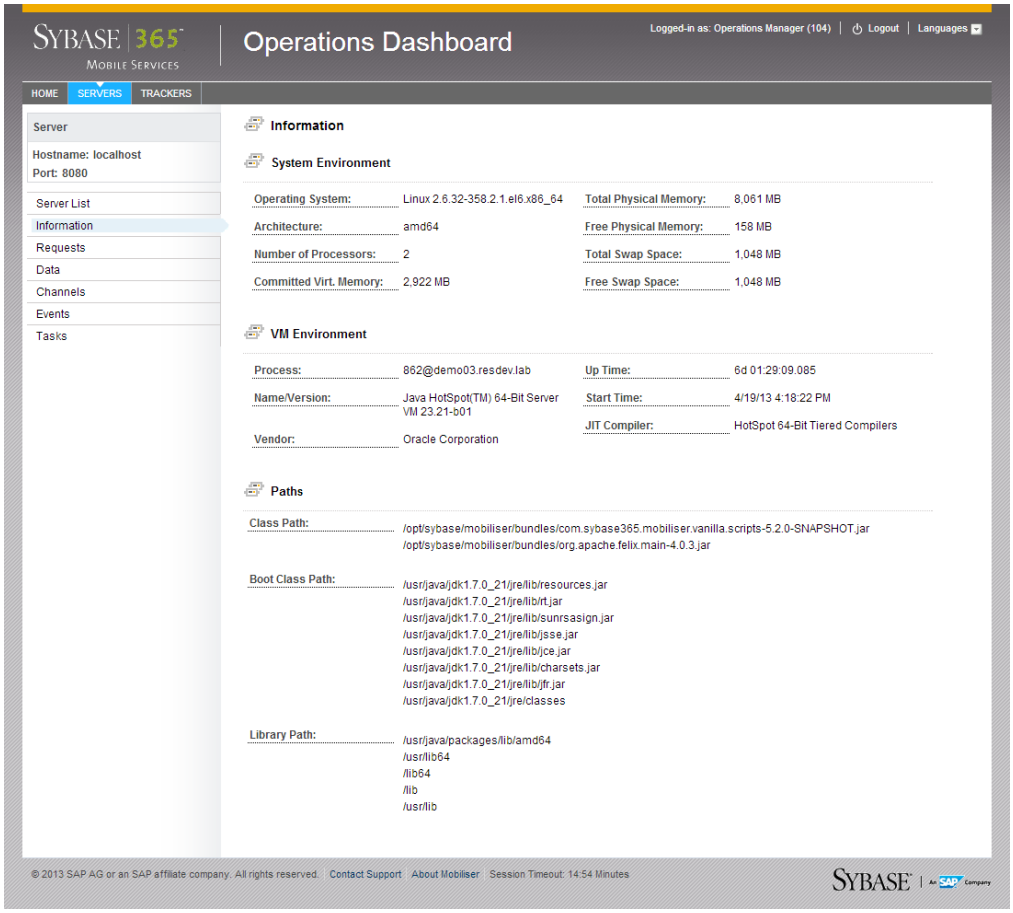
The Server List displays the online and offline servers in the Mobiliser Platform environment. You can select a server that is online to view its information or ping its host. When you select a server, its host name and port display in the left pane. If a server is offline, the visual indicator in the status column shows an orange circle with an exclamation mark. You cannot select or ping a server that is offline.

You can add other servers that are in the Mobiliser Platform environment to the list. To add a server, click **Add a Server** and provide the host name or IP address, and the port number. In addition to adding servers to the list, you can remove servers.



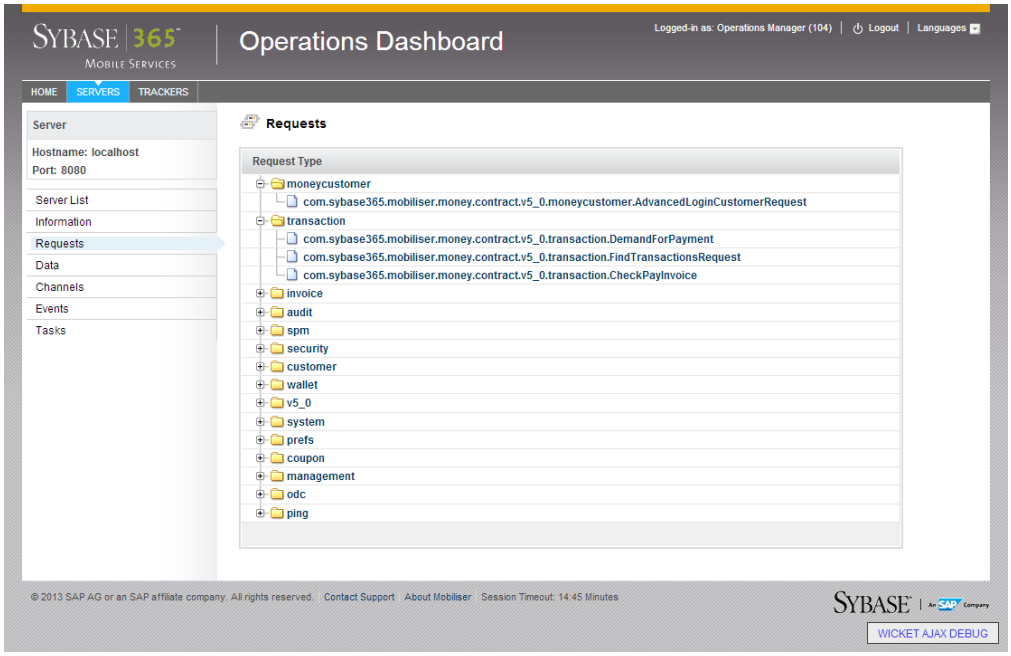
Information

The Information option summarizes the basic system environment information for the selected server, such as number of processors and operating system. You can also view the total and free physical memory, committed virtual memory, and swap space, as well as the time server has been available and class paths.



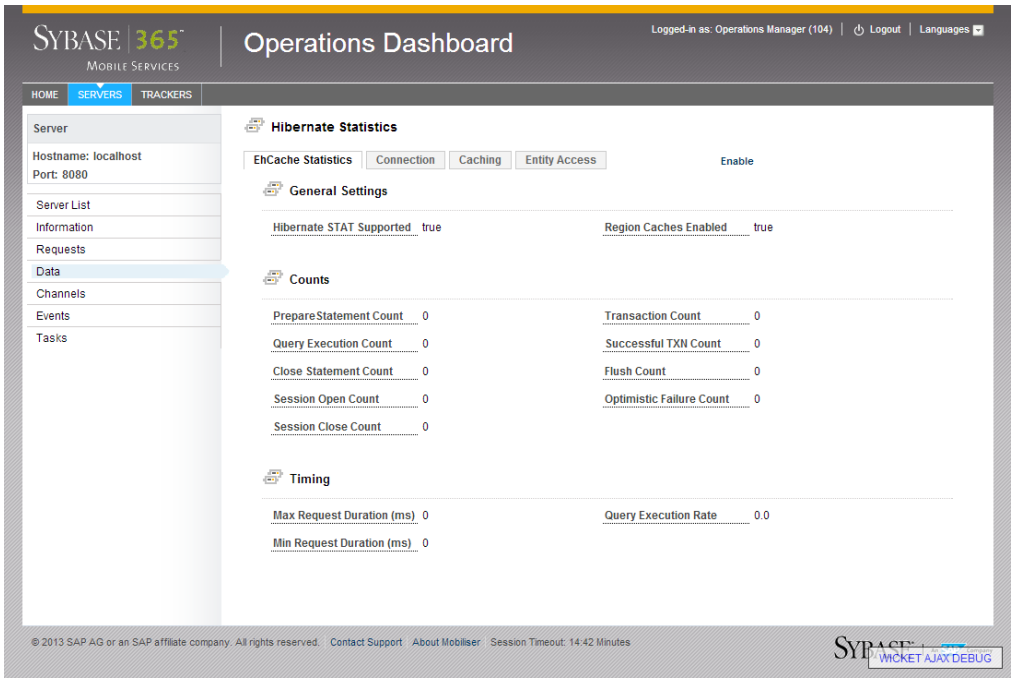
Requests

The Requests option displays all requests made to the Mobiliser Platform server, for example, transaction requests. You can drill down into the statistics of each request. The statistics show the total number of requests made, the success or failure count, and the average response time.



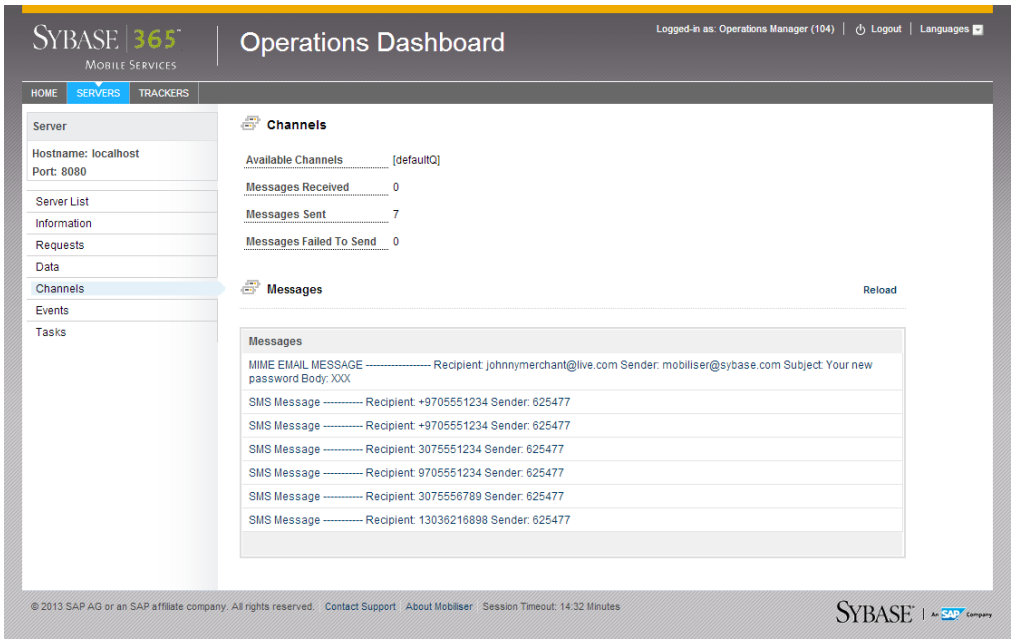
Data

The Data option displays performance monitoring statistics for the selected server, such as EhCache statistics, connection, caching, and entity access. You can enable or disable the performance monitoring.



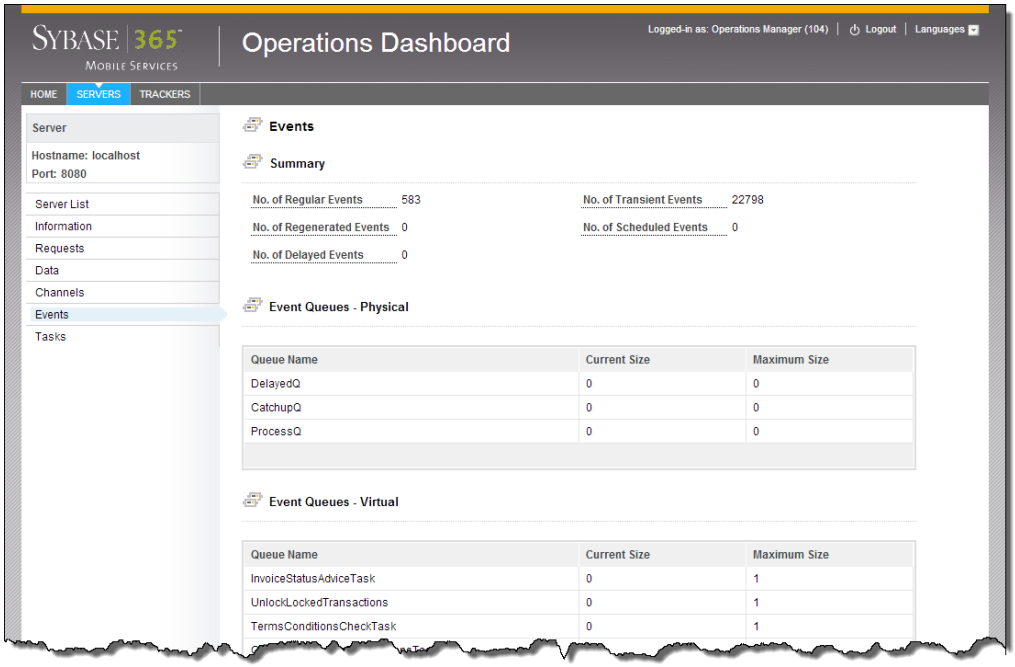
Channels

The Channels option displays the number of messages received, sent, and failed to send. The list shows the last 100 messages that the Mobiliser Platform messaging services generated. You can select a message to view the details, such as date and time stamp. You can also refresh the list to show the most recently sent or received messages.



Events

The Events option displays statistics generated by the Mobiliser Platform event system. The event summary displays the total number of events that the event handlers have generated and processed. An event handler is a procedure that is called when a corresponding event occurs. For an event to be processed by a handler, there has to be an event handler registered for the event name and an available thread from the process pool as determined by the event handler. A single event handler instance is associated with a single event name only.



Event Queues

Event queues display a list of physical and virtual queues, and an instantaneous count of events for the queues.

Table 54. Event Queues

Queue	Description
Physical	Displays the number of events that are present in the physical queue at that instance in time. If the queue is empty, no events are pending for processing.
Virtual	Indicates how many events are in the physical queues for each event name at that instance in time. One virtual queue per event name. If no virtual queues are shown, then no events have been created.

Scheduled Events

Scheduled Events display the internal scheduler system view of all events that are scheduled for triggering. An empty list indicates no known scheduled events.

Table 55. Scheduled Events

Field	Description
Scheduled Event Id	The internal identification of the scheduled event.
Time Zone	An optional time zone in which the cron expression is run.
Cron Expression	An expression conforming to UNIX cron standards for specifying repeats.
End Time	The time beyond which no more triggers fire, or empty if never set.

Field	Description
Start Time	The time the first trigger fired.
Next Fire Time	The expected time of the next trigger fire.
Last Fire Time	The last time the trigger fired.
Trigger	A trigger is a set of criteria that, when met, starts the execution of an event. **Simple - One-off:** Triggers the event once. **Cron - Repeating:** Triggers the event at repeated intervals.

Event Handler

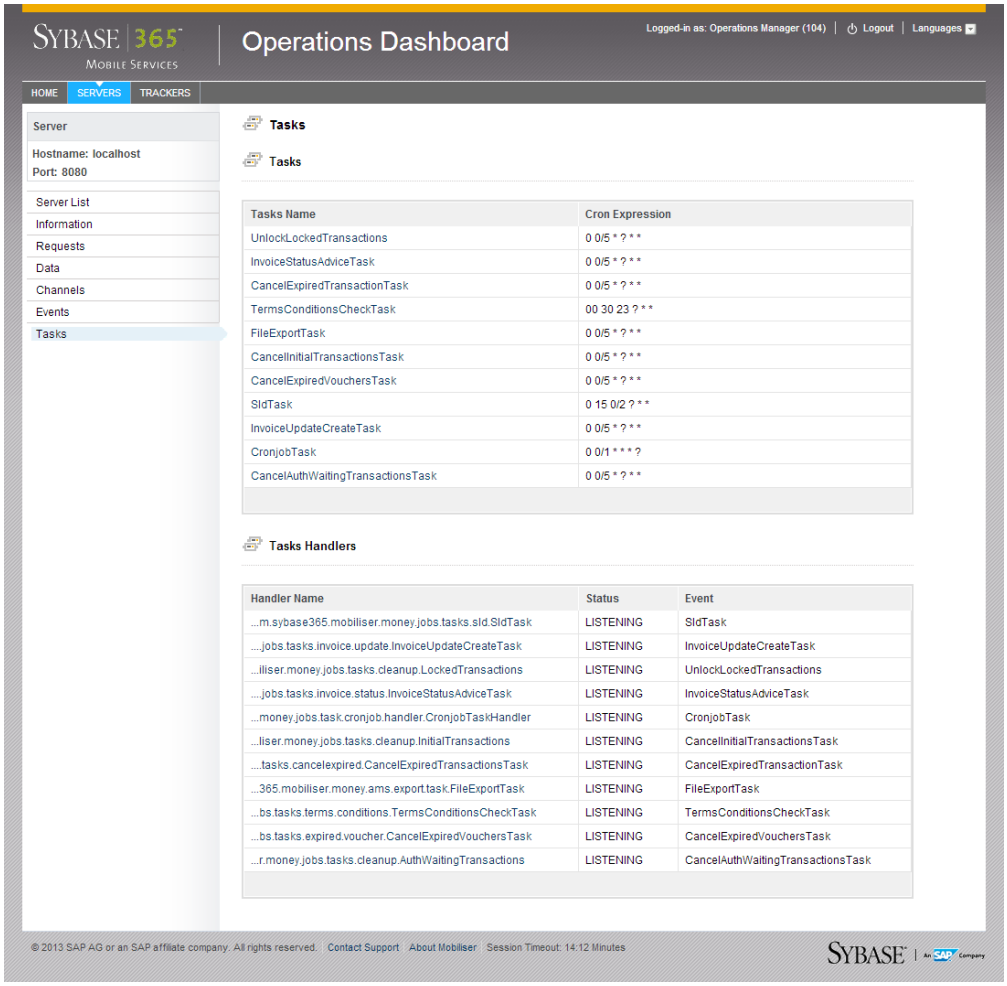
Event handlers display a list of existing handlers registered with the event system.

Table 56. Event Handler

Field	Description
Status	Current status of the event. **Listening:** Active and waiting to be notified when an event occurs. **Catchup:** Regenerated events are still processing.
Event Name	Event name against which the handler is registered.
Current Active Thread	Threads running at this point in time.
Current Idle Thread	Threads allocated to this handler's pool, but not active at this time.
Max Active Thread	Maximum size of event handler thread pool.
Max Idle Thread	Maximum number of idle but not active threads in the pool.
Total Number of Runs	Number of times the task handler was invoked - might be different from Total Events Processed because a handler run does not cause an event to be processed if the handler cannot get a processing status lock on the event or if the event is expired.
Last Run At	Date and time of last run.
Total Events Processed	Number of times the handler process method was called.
Average Process Time (ms)	Average amount of time spent in the handler’s process method.
Total Events Success	Number of events that returned true from its handler process.
Total Events Fail	Number of events that returned false or through an exception from the handler process.
Last Fail At	Date and time of last indicated failed processing event.
Events Marked Expired	Number of events whose expire time has been reached before processing.
Events Marked Catch Up	Number of events still processing regenerated events.

Tasks

The Tasks option displays statistics generated by the Mobiliser Platform event system for tasks as well as the task handlers, which are called by the tasks. Tasks are internal date and time actions scheduled for execution at known repeated intervals. An empty list indicates no known scheduled events. A task is not directly related to an event because a task is not stored in the event system and does not require the regeneration of historical events for a task handler. However, the event system initiates and controls the processing of task actions.



Task Details

Task Details display task information and statistics such as task start time, next and last fire time, and trigger.

Table 57. Task Details

Field	Description
Scheduled Event Id	The internal identification of the scheduled event.
Time Zone	An optional time zone in which the cron expression is run.
Cron Expression	An expression conforming to UNIX cron standards for specifying repeats.
End Time	The time beyond which no more triggers fire, or empty if never set.
Start Time	The time the first trigger fired.
Next Fire Time	The expected time of the next trigger fire.
Last Fire Time	The last time the trigger fired.
Trigger	A trigger is a set of criteria that, when met, starts the execution of an event. **Simple - One-off:** Triggers the event once. **Cron - Repeating:** Triggers the event at repeated intervals.

Task Handlers

Task Handlers display a list of existing task handlers, which are called by the tasks. A task handler coordinates the activities of a task.

Table 58. Task Handlers

Field	Description
Status	Current status of the event. **Listening:** Active and waiting to be notified when an event occurs. **Catchup:** Regenerated events are still processing.
Event Name	Event name against which the handler is registered.
Current Active Thread	Threads running at this point in time.
Current Idle Thread	Threads allocated to this handler's pool, but not active at this time.
Max Active Thread	Maximum size of event handler thread pool.
Max Idle Thread	Maximum number of idle but not active threads in the pool.
Total Number of Runs	Number of times the task handler was invoked - might be different from Total Events Processed because a handler run does not cause an event to be processed if the handler cannot get a processing status lock on the event or if the event is expired.
Last Run At	Date and time of last run.
Total Events Processed	Number of times the handler process method was called.

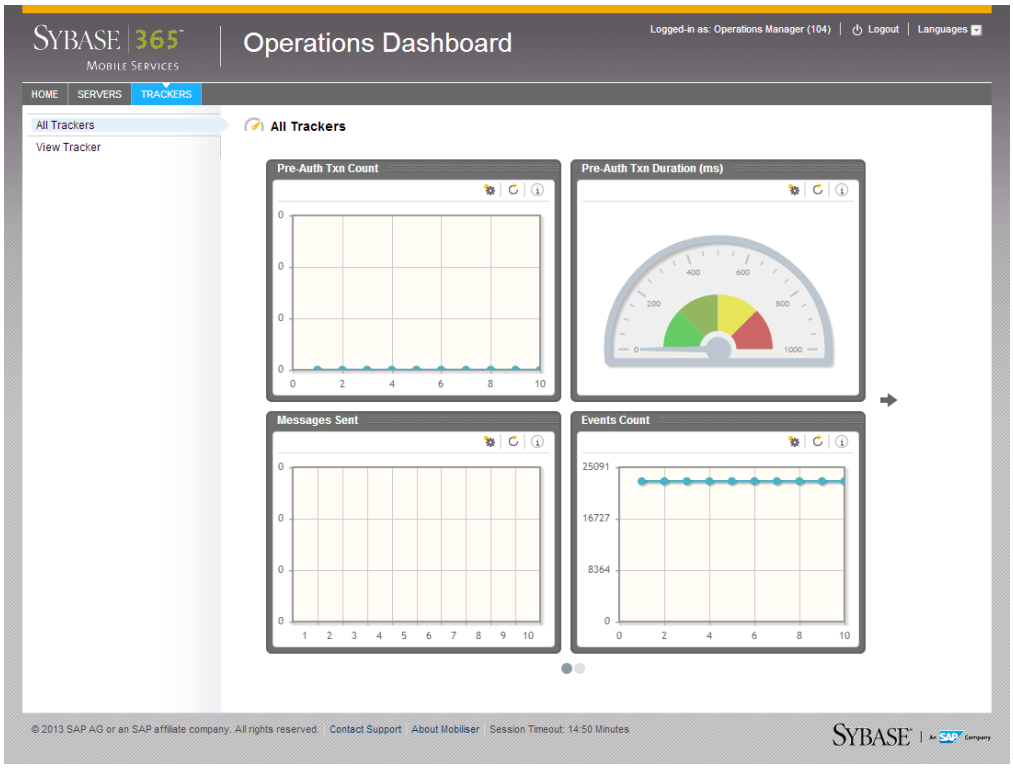
Field	Description
Average Process Time (ms)	Average amount of time spent in the handler's process method.
Total Events Success	Number of events that returned true from its handler process.
Total Events Fail	Number of events that returned false or through an exception from the handler process.
Last Fail At	Date and time of last indicated failed processing event.
Events Marked Expired	Number of events whose expire time has been reached before processing.
Events Marked Catch Up	Number of events still processing regenerated events.

Trackers

The Trackers option lets you visually monitor certain statistics of a Mobiliser Platform system through a series of different chart types such line, bar, and gauge. For example, memory usage is presented as a bar chart, and pre-authorization of a transaction request is presented as a gauge chart.

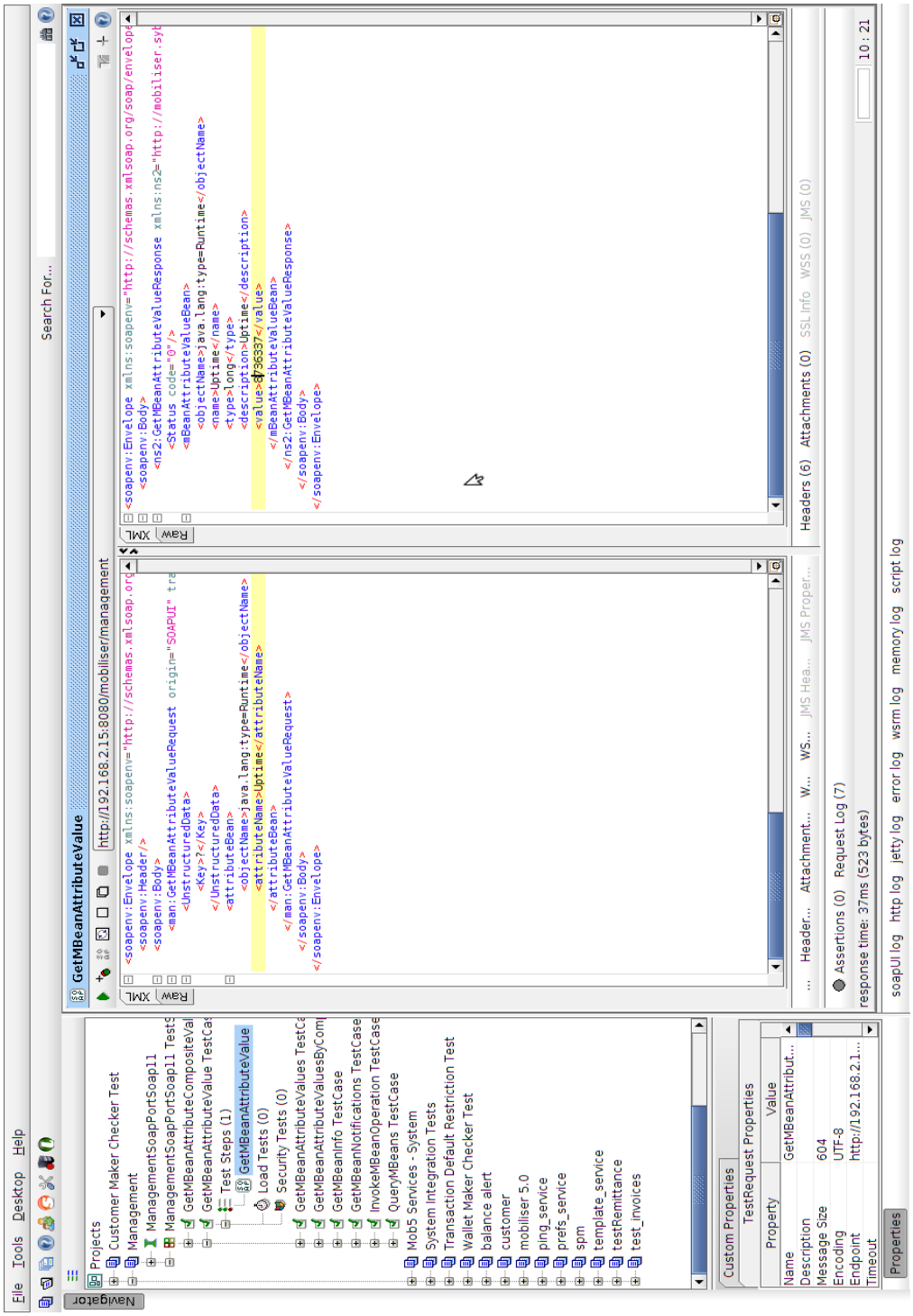
You can monitor and track information such as statistics or status changes from a particular server information point. You can view trackers either on a summary page with easy navigation on the All Trackers page or individually on the View Tracker page.

Note: Trackers can be added via XML configuration by the dashboard developer.



SOAP/REST Interface Management

The Java Management Extensions (JMX) information presented by the Operations Dashboard is accessed through the Mobiliser Platform Management endpoint. This end point translates SOAP requests into requests for local JMX platform objects and attributes information, then sends it back as a SOAP response. The JMX interface can also be accessed via REST returning XML or JSON data.



Tomcat (Web Portals)

Tomcat deploys the Mobiliser Platform UI portals that are used for both administrative and consumer access. According to the Standard Deployment diagram, there are two separate Tomcat instances: one is located on mob-web-1 and the other located on mob-aps-1.

The Tomcat instance located on mob-aps-1 is the internal Web portal. By default, it is configured to connect to the Mobiliser Platform endpoint that allows only the default administrative users to log in.

The administrative users are:

Account	Description
cstfull:secret	Full customer support privileges to the Administration Portal
usermgr:secret	Manage agent accounts
notifmgr:secret	Manage notifications and alerts
headquarter:secret	Create and manage merchants
opsmgr:secret	View and manage system configuration
sysmgr:secret	Monitor all functions of the Mobiliser Platform container

The Tomcat instance located on mob-web-1 is the external Web portal. By default, it is configured to connect to the Mobiliser Platform endpoint that allows consumers to log in.

The external Web portal is the Tomcat instance that is located on mob-web-1. By default, it is configured to connect to the Mobiliser Platform endpoint that allows consumer that originate outside of the Mobiliser Platform (public) to log in.

Changing Tomcat Logging Properties

You can change the log level for various logging appenders.

1. Navigate to the desired Web portal located:
   - **mob-aps-1:** `/opt/sybase/portal/conf`
   - **mob-web-1:** `/opt/sybase/portal/conf`
2. Open the log configuration file `web-ui_log4j.xml`.
3. Change the name/location of the portal log file and change the log level for various logging appenders:

```
<log4j:configuration
xmlns:log4j="http://jakarta.apache.org/log4j/">
  <appender name="FILE"
class="org.apache.log4j.DailyRollingFileAppender">
    <param name="File" value="${log4j.logfiles.path}/web-
ui.log" />
    <param name="Append" value="true" />
```

```
    <param name="DatePattern" value="'.'yyyy-MM-dd" />
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%d{ISO8601} [%t]
%-5p %c:%L %x - %m%n" />
    </layout>
  </appender>
  <logger name="com.sybase365">
    <level value="TRACE" />
  </logger>
  <logger name="com.sybase365.mobiliser.util">
    <level value="WARN" />
  </logger>
  <logger name="org.apache">
    <level value="INFO" />
  </logger>
  <logger name="org.apache.wicket.util.resource">
    <level value="WARN" />
  </logger>
  <root>
    <level value="WARN" />
    <appender-ref ref="FILE" />
  </root>
</log4j:configuration>
```

Data Archiving, Retention and Deletion

Currently, Mobiliser Platform does not support data archiving, data retention, or deletion policies implemented. Therefore, it is important to use default database technology to implement any desired procedures and policies.

Data Archiving

You can move transactional data from the online transactional database safely. However, it is not recommended to move customer data since this information is required in the transactional online database to ensure referential integrity. The customer data should be small compared to the amount of transactional data in the system. When archiving data from the online database, the data is no longer visible through the standard Mobiliser Platform user interfaces.

Note: Be aware of foreign key constraints when moving transactional data.

Make sure to move all information associated to a transaction, including the audit, history, and traceable request data:

- MOB_TXNS
- MOB_SUB_TXNS.ID_TXN->MOB_TXNS.ID_TXN
- MOB_TXN_ATTRIBUTES.ID_TXN->MOB_TXNS.ID_TXN
- MOB_FEES.ID_SUB_TXN->MOB_SUB_TXNS.ID_SUB_TXN
- MOB HISTORY (tracks changes to individual columns in the database)
- MOB_AUDIT_LOGS (tracks each remote service call)
- MOB_TRACEABLE_REQUESTS (stores 24 hours of data for non-repudiation and response dehydration)

If the transaction is an invoice payment, you must also move the invoice information:

- MOB_INVOICES
- MOB_INV_TXNS.ID_TXN->MOB_TXNS.ID_TXN
- MOB_INV_TXNS.ID_INVOICE->MOB_INVOICES.ID_INVOICE
- MOB_INV_ATTRIBUTES.ID_INVOICE->MOB_INVOICES.ID_INVOICE

Data Retention and Deletion

Mobiliser Platform does not have automated procedures to implement data retention and deletion policies. Therefore, you can setup cron jobs or manually perform tasks to delete data after the retention period has expired. Since the Mobiliser Platform database holds a number of referential integrity constraints binding a customer record to transactions and other entities, you must encrypt the customer data instead of physically deleting it. That is, to delete the customer record from the system any personally identifiable information (PII) must be overwritten with random text, for example DELETED.

Customer data is stored in the following tables:

- MOB_CUSTOMERS
- MOB_CUSTOMERS_IDENTIFICATIONS.ID_CUSTOMER->MOB_CUSTOMER.ID_CUSTOMER
- MOB_CUSTOMERS_CREDENTIALS.ID_CUSTOMER->MOB_CUSTOMER.ID_CUSTOMER
- MOB_CUSTOMERS_IDENTITIES.ID_CUSTOMER->MOB_CUSTOMER.ID_CUSTOMER
- MOB_CUSTOMERS_ATTRIBUTES.ID_CUSTOMER->MOB_CUSTOMER.ID_CUSTOMER
- MOB_ADDRESSES.ID_CUSTOMER->MOB_CUSTOMER.ID_CUSTOMER
- MOB_ATTACHMENTS.ID_CUSTOMER->MOB_CUSTOMER.ID_CUSTOMER
- MOB_NOTES.ID_CUSTOMER->MOB_CUSTOMER.ID_CUSTOMER
- MOB_PIS.ID_CUSTOMER->MOB_CUSTOMER.ID_CUSTOMER
- MOB_PIS.ID_PI<-MOB_WALLET->MOB_CUSTOMER.ID_CUSTOMER
- MOB_SVA.ID_PI->MOB_PIS.ID_PI
- MOB_CREDIT_CARDS.ID_PI->MOB_PIS.ID_PI
- MOB_BANK_ACCOUNTS.ID_PI->MOB_PIS.ID_PI
- MOB_EXTERNAL_ACCOUNTS.ID_PI->MOB_PIS.ID_PI

Note: Customized projects may introduce more tables holding PII.

Deletion Script

Use the deletion script to remove and deactivate customer related information such as mobile phone numbers, passwords, PINs, and payment instruments. The deletion script also removes and marks customers as inactive in the system. Attachments, notes, and change history of the customer is also removed. Therefore, further processing of financial transactions is not possible.

```
-- delete information about bank accounts

UPDATE MOB_BANK_ACCOUNTS SET STR_NAME = '###', STR_NAME_BANK = '###',
STR_CITY_BANK = '###', STR_INSTITUTION_CODE = '###',

STR_BRANCH_CODE = '###', STR_ACCOUNT_NUMBER = '###', STR_DISPLAY_NUMBER
= '###' WHERE

ID_PI in (SELECT ID_PI FROM MOB_PIS WHERE ID_CUSTOMER = ?);

-- delete information about "other" financial accounts

UPDATE MOB_EXTERNAL_ACCOUNTS SET STR_ID1 = '###', STR_ID2 = '###',
STR_ID3 = '###', STR_ID4 = '###',

STR_ID8 = '###', STR_ID7 = '###', STR_ID6 = '###', STR_ID5 = '###' WHERE

ID_PI in (SELECT ID_PI FROM MOB_PIS WHERE ID_CUSTOMER = ?);

-- delete credit card information
```

```
UPDATE MOB_CREDIT_CARDS SET STR_CARD_NUMBER = '###',
STR_CARD_HOLDER_NAME = '###', STR_DISPLAY_NUMBER = '###' WHERE

ID_PI in (SELECT ID_PI FROM MOB_PIS WHERE ID_CUSTOMER = ?);

-- delete names of accounts

UPDATE MOB_WALLET SET STR_ALIAS = '###' WHERE ID_CUSTOMER = ?;

-- mark all accounts as inactive

UPDATE MOB_PIS SET BOL_IS_ACTIVE = 'N' WHERE ID_CUSTOMER = ?;

-- delete all identifications, such as mobile phone number and make them
inactive

UPDATE MOB_CUSTOMERS_IDENTIFICATIONS SET STR_IDENTIFICATION = '###',
BOL_IS_ACTIVE = 'N' WHERE ID_CUSTOMER = ?;

-- delete all passwords and PINs and make them inactive

UPDATE MOB_CUSTOMERS_CREDENTIALS SET STR_CREDENTIAL = '###',
BOL_IS_ACTIVE = 'N' WHERE ID_CUSTOMER = ?;

-- delete all identity (e.g. passport) information

UPDATE MOB_CUSTOMERS_IDENTITIES SET STR_IDENTITY = '###',
STR_ISSUE_PLACE = '###', STR_ISSUER = '###', BOL_IS_ACTIVE = 'N' WHERE
ID_CUSTOMER = ?;

-- delete all general purpose attributes

UPDATE MOB_CUSTOMERS_ATTRIBUTES SET STR_VALUE = '###' WHERE ID_CUSTOMER
= ?;

-- delete all binary attachments

UPDATE MOB_ATTACHMENTS SET STR_NAME = '###', BIN_CONTENT = null WHERE
ID_CUSTOMER = ?;

-- delete all notes (system generated or manually entered)

UPDATE MOB_NOTES SET STR_SUBJECT = '###', STR_TEXT = '###' WHERE
ID_CUSTOMER = ?;

-- mark customer as inactive

UPDATE MOB_CUSTOMERS SET STR_DISPLAY_NAME = '###', STR_SECURITY_QUESTION
= '###', STR_SECURITY_ANSWER = '###', STR_REFERRAL_CODE = '###',
BOL_IS_ACTIVE = 'N', DAT_DATE_OF_BIRTH = null WHERE ID_CUSTOMER = ?;

-- delete all address information

UPDATE MOB_ADDRESSES SET STR_FIRST_NAME = '###', STR_MIDDLE_NAME =
'###', STR_LAST_NAME = '###', STR_TITLE = '###', STR_COMPANY1 = '###',
STR_COMPANY2 = '###',

STR_COMPANY_SHORTNAME = '###', STR_POSITION = '###', STR_STREET1 =
'###', STR_STREET2 = '###', STR_HOUSE_NUMBER = '###', STR_ZIP = '###',
STR_CITY = '###',

STR_STATE = '###', STR_PHONE1 = '###', STR_PHONE2 = '###', STR_FAX =
'###', STR_EMAIL = '###', STR_URL = '###', STR_NAME_ADDRESS = '###'
WHERE ID_CUSTOMER = ?;

-- delete all information regarding change history
```

```
UPDATE MOB_HISTORY SET STR_OLD_VALUE = '###', STR_NEW_VALUE = '###'
WHERE ID_OBJECT = ?;

UPDATE MOB_HISTORY SET STR_OLD_VALUE = '###', STR_NEW_VALUE = '###'
WHERE ID_OBJECT in (SELECT ID_PI FROM MOB_PIS WHERE ID_CUSTOMER = ?);

UPDATE MOB_HISTORY SET STR_OLD_VALUE = '###', STR_NEW_VALUE = '###'
WHERE ID_OBJECT in (SELECT ID_ADDRESS FROM MOB_ADDRESSES WHERE
ID_CUSTOMER = ?);

UPDATE MOB_HISTORY SET STR_OLD_VALUE = '###', STR_NEW_VALUE = '###'
WHERE ID_OBJECT in (SELECT ID_IDENTITY FROM MOB_CUSTOMERS_IDENTITIES
WHERE ID_CUSTOMER = ?);

UPDATE MOB_HISTORY SET STR_OLD_VALUE = '###', STR_NEW_VALUE = '###'
WHERE ID_OBJECT in (SELECT ID_CUSTOMER_IDENTIFICATION FROM
MOB_CUSTOMERS_IDENTIFICATIONS WHERE ID_CUSTOMER = ?);

UPDATE MOB_HISTORY SET STR_OLD_VALUE = '###', STR_NEW_VALUE = '###'
WHERE ID_OBJECT in (SELECT ID_CUSTOMER_CREDENTIAL FROM
MOB_CUSTOMERS_CREDENTIALS WHERE ID_CUSTOMER = ?);

UPDATE MOB_HISTORY SET STR_OLD_VALUE = '###', STR_NEW_VALUE = '###'
WHERE ID_OBJECT in (SELECT ID_NOTE FROM MOB_NOTES WHERE ID_CUSTOMER =
?);

-- commit all data

commit;
```

SAP Interoperability

In an interconnected SAP-driven computing ecosystem, enabling interoperability between products is an important operations management task that enables Mobiliser Platform runtime data sharing with these SAP systems.

SAP System Landscape Directory Server Overview

For environments that use SAP® Solution Manager for runtime root-cause analysis, use the built-in system landscape directory (SLD) data supplier to send the configured SLD data periodically to Solution Manager. This configuration allows Mobiliser Platform to deliver runtime information to a common SLD repository, keeping information about your Mobiliser Platform infrastructure complete and current.

SLD Payload Configuration

Mobiliser Platform ships with a template configuration document in:

```
/opt/sybase/money/conf/sld_template.xml
```

This document should be configured with the relevant SLD details. You can use placeholders in the form of `${key}`. The values used for replacement are searched for in the following order:

1. system properties (`-D` parameter supplied when starting the JVM)
2. in the property file:
   `/opt/sybase/money/conf/cfgbackup/com.sybase365.mobiliser.money.jobs.tasks.sld.properties`
3. One of the following special keys, if not already set in system properties or the property file:
   a. **fqdn** – is replaced with `InetAddress.getLocalHost().getHostName()`
   b. **ip** – is replaced with `InetAddress.getLocalHost().getHostAddress()`
   c. **host name** – is replaced with `InetAddress.getLocalHost().getHostName()`

You can also modify the value for `installation.id` to specify the name of the installation. The default value is VANILLA.

SLD Transfer Configuration

The server and access credential must be configured using the Mobiliser Platform preferences service in the Operations Dashboard to transfer SLD successfully.

The SLD information is submitted by SldTask shortly after starting the container and then again following the configured cron pattern.

Path	/businesslayer/system/com/sybase365/mobiliser/money/jobs/tasks/sld/SldTask/

Key	Description
sld.active	Indicates of the supplier is enabled or not. The default is: `false`
sld.username	Defines the user for uploading the SLD XML data.
sld.password	Defines the password used for uploading the SLD XML data.
sld.url	Defines the URL used for uploading the SLD XML data. The default is: http://localhost:50100/sld/ds
task.cronpattern	Defines the cron pattern used to specify the upload schedule. The default is: `0 15 0/12 ? * *`
sld.resource	Defines the SLD template resource. The default is: `${mobiliser.home}/conf/sld_template.xml`

Managed System Setup of the Portal (Tomcat)

SAP Solution Manager 7.1 supports Apache Tomcat products with end-to-end root cause analysis. The managed system setup is run similar to other applications on Tomcat:

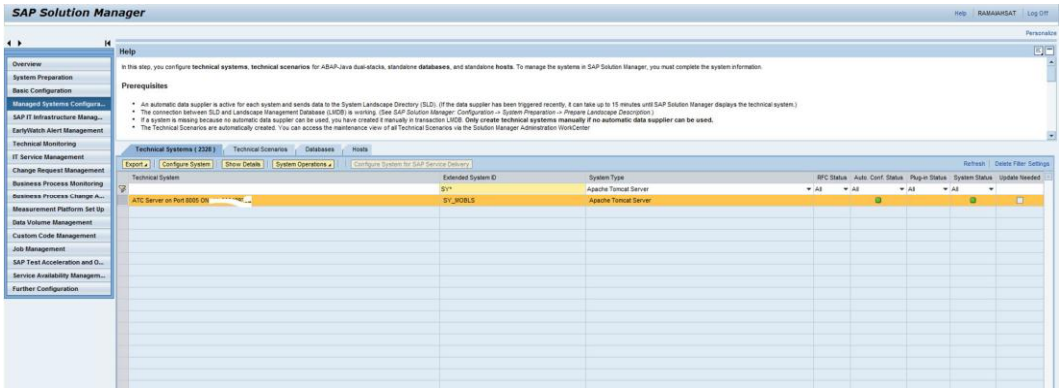
http://wiki.sdn.sap.com/wiki/display/SMSETUP/Managed+System+Setup+of+Apache+Tomcat+System+in+Solman+7.1

Running SLD Integration

1. Refer to SAP Note 1508421 SAP Solution Landscape Directory (SLD) Data Supplier Integration Details for Apache Tomcat.
2. Place the TOMCAT SLD Data Supplier:

   `C:\<SybaseMobiliser_TomCat_Installation_Path>\com.sybase365.mobiliser.dist.oracle-<version>\web\webapps`
3. Place the SAP_Metadata.xml in:

   (specific to Sybase Mobiliser Platform Release attached to this Note)
   `C:\<SybaseMobiliser_TomCat_Installation_Path>\com.sybase365.mobiliser.dist.oracle-<version>\web\webapps\portal\META-INF`
4. Refer to SAP Note 1438005 for Setup of Tomcat Integration with Wily Introscope.
5. Restart the Sybase Mobiliser Platform Tomcat Portal.
6. Check the Tomcat and Introscope agent logs to confirm the SLD registration and Introscope integration with Mobiliser Platform successfully completed.
7. Run Managed System Configuration.
8. Select **Sybase Mobiliser System**.


SAP Solution Manager
Help
Prerequisites
Technical Systems (2328)
Technical Scenarios
Databases
Hosts
Export
Configure System
Show Details
System Operations
Technical System
Extended System ID
System Type
ATC Server on Port 8005 ON
SY_WOBLS
Apache Tomcat Server

Cron Expression Reference

A cron expression is a string comprised of six or seven fields separated by white space. Fields can contain any of the allowed values, along with various combinations of the allowed special characters for that field. Cron expressions can be as simple as * * * * ? * or as complex as 0/5 14,18,3-39,52 * ? JAN,MAR,SEP MON-FRI 2002-2010.

Fields can contain any of the allowed values alone with various combinations of the allowed special characters for that field.

Table 59. Cron Expression Format

Field Name	Allowed Value	Allowed Special Characters
Seconds	0-59	, - * /
Minutes	0-59	, - * /
Hours	0-23	, - * /
Day-of-Month	1-31	, - * ? / L W
Month	1-12 or JAN-DEC	, - * /
Day-of-Week	1-7 or SUN-SAT	, - * ? / L #
Year (Optional)	empty, 1970-2199	, - * /

Table 60. Special Characters

Character	Description
*	Asterisks indicate that the cron expression matches for all values of the field. For example, "*" in the minute field means every minute.
?	Question marks are used to specify ‘no specific value’ and are allowed for the day-of-month and day-of-week fields. It is used instead of the asterisk (*) for leaving either day-of-month or day-of-week blank.
-	Hyphens are used to define ranges. For example, "10-12" in the hour field means the hours of 10, 11, and 12.
,	Commas are used to separate items of a list. For example, "MON,WED,FRI" in the day-of-week field means the days Monday, Wednesday, and Friday.
/	Forward slash are used to indicate increments. For example. "0/15" in the seconds field means the seconds 0, 15, 30, and 45. Additionally, "1/3" in the day-of-month field means every 3 days starting on the first day of the month.
L	Short-hand for "last" and is allowed for the day-of-month and day-of-week fields. The "L" character has a different meaning in each of the two fields. For example, "L" in the day-of-month field means the last day of the month. If used in the day-of-week field, it means 7 or SAT. However, if used in the day-of-week field after another value, it means the last xxx day of the month. For example, "6L" in the day-of-week field means the last Friday of the month.
W	Short-hand for "weekday" and is allowed for the day-of-month field. The "W"

Character	Description
	character is used to specify the weekday nearest the given day. For example, "15W" in the day-of-month field means the nearest weekday to the 15th of the month. Therefore, if the 15th is a Saturday, the job runs on Friday the 14th. The "L" and "W" characters can be combined in the day-of-month field. For example, "LW" means the last weekday of the month.
#	Hash marks specify constructs. For example, "6#3' in the day-of-week field means the third Friday of the month.

Table 61. Cron Expression Examples

Expression	Description
0 0 12 * * ?	Triggered to run at 12:00 p.m. (noon) every day
0 15 10 ? * *	Triggered to run at 10:15 a.m. every day
0 15 10 * * ?	Triggered to run at 10:15 a.m. every day
0 15 10 * * ? *	Triggered to run at 10:15 a.m. every day
0 15 10 * * ? 2005	Triggered to run at 10:15 a.m. every day during the year 2005
0 * 14 * * ?	Triggered to run every minute starting at 2:00 p.m. and ending at 2:59 p.m., every day
0 0/5 14 * * ?	Triggered to run every 5 minutes starting at 2:00 p.m. and ending at 2:55 p.m., every day
0 0/5 14,18 * * ?	Triggered to run every 5 minutes starting at 2:00 p.m. and ending at 2:55 p.m., AND fire every 5 minutes starting at 6:00 p.m. and ending at 6:55 p.m., every day
0 0-5 14 * * ?	Triggered to run every minute starting at 2:00 p.m. and ending at 2:05 p.m., every day
0 10,44 14 ? 3 WED	Triggered to run at 2:10 p.m. and at 2:44 p.m. every Wednesday in the month of March
0 15 10 ? * MON-FRI	Triggered to run at 10:15 a.m. every Monday, Tuesday, Wednesday, Thursday and Friday
0 15 10 15 * ?	Triggered to run at 10:15 a.m. on the 15th day of every month
0 15 10 L * ?	Triggered to run at 10:15 a.m. on the last day of every month
0 15 10 L-2 * ?	Triggered to run at 10:15 a.m. on the 2nd-to-last last day of every month
0 15 10 ? * 6L	Triggered to run at 10:15 a.m. on the last Friday of every month
0 15 10 ? * 6L 2002-2005	Triggered to run at 10:15 a.m. on every last Friday of every month during the years 2002, 2003, 2004 and 2005
0 15 10 ? * 6#3	Triggered to run at 10:15 a.m. on the third Friday of every month
0 0 12 1/5 * ?	Triggered to run at 12:00 p.m. (noon) every 5 days every month, starting on the first day of the month

Expression	Description
0 11 11 11 11 ?	Triggered to run every November 11 at 11:11 a.m.

Troubleshooting

Issue: DBMaintain command is not executing

Error:

```
**************************************************************

# Error:Unable to connect to database. Driver class not found:
oracle.jdbc.driver.OracleDriver

# Cause:oracle.jdbc.driver.OracleDriver

**************************************************************
```

Explanation/Resolution: Download the JDBC driver for the database you are attempting to run DBMaintain on. Specify the path/location of the JDBC driver in the dbmaintain.properties.<database> file. The variable that needs to be updated is database.driverLocation.

Issue: DBMaintain command is not executing

Error:

```
**************************************************************

# Error:Unable to connect to database. Could not create
connection for database url:
jdbc:oracle:thin:@localhost:1521:xe, user name: mobr5, password:
<not shown>

# Cause:Listener refused the connection with the following
error:

ORA-12505, TNS:listener does not currently know of SID given in
connect descriptor

**************************************************************
```

Explanation/Resolution: The JDBC URL, which specifies the location of your database, is either incorrect or is not reachable from the localhost being used. Update the `datbase.url` in the dbmaintain.properties.<database> file to the correct value or resolve any security policies that may block communication from the localhost to the database URL.

Issue: Mobiliser Container will not initialize

mobiliser.log Error:

```
2013-09-17 22:19:02,210 [main] WARN  org.activiti.osgi.Activator
- FileInstall package is not available, disabling fileinstall
support
2013-09-17 22:19:03,069 [main] WARN
org.eclipse.gemini.blueprint.extender.internal.support.Namespace
Plugins - Bundle AIMS Mobiliser :: Framework :: Service
```

```
Configuration (com.sybase365.mobiliser.framework.service.config)
cannot see class
[org.springframework.beans.factory.xml.NamespaceHandlerResolver]
; ignoring it as a namespace resolver
2013-09-17 22:19:03,973 [main] WARN
org.eclipse.gemini.blueprint.extender.internal.support.Namespace
Plugins - Bundle AIMS Mobiliser :: Framework :: Service
Configuration (com.sybase365.mobiliser.framework.service.config)
cannot see class
[org.springframework.beans.factory.xml.NamespaceHandlerResolver]
; ignoring it as a namespace resolver
2013-09-17 22:19:18,979 [aims-init-14] WARN
com.sybase365.mobiliser.framework.vscan.scanner.impl.VScanImpl -
Cannot initialize Virus Scan Service. The following service
exception occured: Expecting an absolute path of the library:
libsavisap.so
2013-09-17 22:21:33,180 [aims-init-36] WARN
net.sf.ehcache.hibernate.strategy.EhcacheAccessStrategyFactoryIm
pl - read-only cache configured for mutable entity
[com.sybase365.mobiliser.money.ams.dao.model.CurrencyExchange]
2013-09-17 22:21:33,523 [aims-init-36] WARN
net.sf.ehcache.hibernate.strategy.EhcacheAccessStrategyFactoryIm
pl - read-only cache configured for mutable entity
[com.sybase365.mobiliser.money.ams.dao.model.ClearingConfig]
2013-09-17 22:25:31,762 [aims-init-54] WARN
com.sybase365.mobiliser.money.businesslogic.billpayment.impl.Bil
lPaymentBrokerImpl - Replaced BillPaymentHandler [$Proxy366]
with [Standard Offline Billpayment Handler] for coverage [2].
2013-09-17 22:25:31,763 [aims-init-54] WARN
com.sybase365.mobiliser.money.businesslogic.billpayment.impl.Bil
lPaymentBrokerImpl - Replaced BillPaymentHandler [$Proxy366]
with [No-Op BillPayment Handler] for coverage [3].
2013-09-17 22:29:27,231 [Spring DM Context Creation Timer] WARN
org.eclipse.gemini.blueprint.extender.internal.dependencies.star
tup.DependencyWaiterApplicationContextExecutor - Timeout
occurred before finding service dependencies for
[OsgiBundleXmlApplicationContext(bundle=com.sybase365.mobiliser.
util.messaging.channelmanager.engine.dummyreceiver,
config=osgibundle:/META-INF/spring/*.xml)]
2013-09-17 22:29:27,232 [Spring DM Context Creation Timer] ERROR
org.eclipse.gemini.blueprint.extender.internal.activator.Context
LoaderListener - Application context refresh failed
(OsgiBundleXmlApplicationContext(bundle=com.sybase365.mobiliser.
util.messaging.channelmanager.engine.dummyreceiver,
config=osgibundle:/META-INF/spring/*.xml))
org.springframework.context.ApplicationContextException:
Application context initialization for
'com.sybase365.mobiliser.util.messaging.channelmanager.engine.du
mmyreceiver' has timed out waiting for
```

```
(objectClass=com.sybase365.mobiliser.util.messaging.template.api
.IMessagingEngine)
```

felix.out Error:

```
Welcome to Apache Felix Gogo^M

^M

Persistence bundle starting...

Persistence bundle started.

Auto-deploy start: org.osgi.framework.BundleException:
Unresolved constraint in bundle
com.sybase365.mobiliser.util.prefs.cm [441]: Unable to resolve
441.0: missing requirement [441.0] osgi.wiring.package;
(&(osgi.wiring.package=com.sybase365.mobiliser.util.prefs.api)(v
ersion>=5.1.0)(!(version>=6.0.0)))

Auto-deploy start: org.osgi.framework.BundleException:
Unresolved constraint in bundle
com.sybase365.mobiliser.util.prefs.impl [447]: Unable to resolve
447.0: missing requirement [447.0] osgi.wiring.package;
(&(osgi.wiring.package=com.sybase365.mobiliser.util.prefs.api)(v
ersion>=5.1.0)(!(version>=6.0.0)))

Auto-deploy start: org.osgi.framework.BundleException:
Unresolved constraint in bundle
com.sybase365.mobiliser.util.prefs.management.jmx [449]: Unable
to resolve 449.0: missing requirement [449.0]
osgi.wiring.package;
(&(osgi.wiring.package=com.sybase365.mobiliser.util.prefs.api)(v
ersion>=5.1.0)(!(version>=6.0.0)))

Auto-deploy start: org.osgi.framework.BundleException:
Unresolved constraint in bundle
com.sybase365.mobiliser.util.prefs.store.api [452]: Unable to
resolve 452.0: missing requirement [452.0] osgi.wiring.package;
(&(osgi.wiring.package=com.sybase365.mobiliser.util.prefs.api)(v
ersion>=5.1.0)(!(version>=6.0.0)))

Auto-deploy start: org.osgi.framework.BundleException:
Unresolved constraint in bundle
com.sybase365.mobiliser.util.prefs.store.local [453]: Unable to
resolve 453.0: missing requirement [453.0] osgi.wiring.package;
(&(osgi.wiring.package=com.sybase365.mobiliser.util.prefs.api)(v
ersion>=5.1.0)(!(version>=6.0.0)))

Auto-deploy start: org.osgi.framework.BundleException:
Unresolved constraint in bundle
com.sybase365.mobiliser.util.prefs.util [454]: Unable to resolve
454.0: missing requirement [454.0] osgi.wiring.package;
(&(osgi.wiring.package=com.sybase365.mobiliser.util.prefs.api)(v
ersion>=5.1.0)(!(version>=6.0.0)))
```

Other Behaviors: The bundles in the `felix.out` log file may take a long time to load and eventually times out.

Explanation/Resolution: The required jar file needed for the Mobiliser container was not copied to the proper directory as described in the "Install/Copy Preferences Software (mob-aps-1)" section in the Mobiliser Platform Installation Guide 5.1 SP03. Please revisit that section and copy the required jar file to the proper location.

Issue: Unable to reach Mobiliser Customer WSDL

Browser Error:

```
HTTP ERROR: 404

Problem accessing /mobiliser/customer/Customer.wsdl. Reason:

Not Found
```

mobiliser.log Error:

```
2013-04-19 19:33:48,036 [aims-init-5] WARN
org.springframework.jdbc.datasource.LazyConnectionDataSourceProx
y - Could not retrieve default auto-commit and transaction
isolation settings

java.sql.SQLException: ------

java.lang.ClassNotFoundException: oracle.jdbc.OracleDriver not
found by com.jolbox.bonecp [28]
```

Other Behaviors: The bundles in the `felix.out` log file may take a long time to load and eventually times out.

Explanation/Resolution: The JDBC JAR file that needs to be created and added to `{MOBILISER_HOME}/bundles/07-frameworks`, was either not created or it was renamed incorrectly. The content of everything that is needed to create this vital JAR file is located in:

- **Oracle:** `/applications/oracle`
- **DB2:** `/applications/ibm`

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

W