# MCA Services Configuration and Administration Guide

Version 2005, Rev. A

June 2005

Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404

# Contents

# 1 What's New in This Release

## What's New in MCA Services Configuration and Administration Guide, Version 2005, Rev. A

Table 2 lists changes in this version of the documentation to support release 2005 of the software.

Table 1. What's New in MCA Services Configuration and Administration Guide, Version 2005, Rev. A

| Topic | Description |
|---|---|
| Prerequisites for Configuring TestCustomerData.properties, page 11 | A section has been added on the prerequisites for configuring TestCustomerData.properties. |
| Configuring this.absolutePath, page 11 | A section has been added on configuring the `this.absolutepath` property. |
| Prerequisites for Configuring BankframeResource.properties, page 13 | A section has been added on the prerequisites for configuring BankframeResource.properties. |

## What's New in MCA Services Configuration and Administration Guide, Version 2005

Table 2 lists changes in this version of the documentation to support release 2005 of the software.

Table 2. What's New in MCA Services Configuration and Administration Guide, Version 2005

| Topic | Description |
|---|---|
| HTTPSClient Settings, page 14 | The setting `channel.https.ssl.protocol` has been added. This setting determines the SSL protocol class that is used in the `java.protocol.handler.pkgs` system property.<br><br>The setting `channel.https.ssl.provider` has been added. This setting determines the JSSE (Java Secure Socket Extension) SSL provider class that is used in conjunction with channel.https.ssl.protocol. |
| Console Logger Setting, page 17 | The `console.logger` key has been added to enable the configuration of debug logging messages in the generic console logging framework. Turning off console logger debug messages reduces output and improves performance. The syntax is |

| Topic | Description |
|---|---|
| | `console.logger=<Logging level to use>`. |
| LDAP Context Cache Setting, page 20 | The `Ldap.context.cache` key has been added to make the server context caching configurable. This setting only needs to be configured if you want to enable caching. If this property is not present in the `BankFrameResource.properties` file, the value defaults to `False`. |
| Caching Framework Settings, page 27 | The `cache.index.<INDEX_NAME>` key has been added to specify the name of a cache index and the `CacheIndexer` class for managing the cache index. If there is no entry in the `BankframeResource.properties` file for INDEX_NAME, the `CacheIndexFactory` returns an instance of `CacheIndex` by default. If a specific `CacheIndexer` is not defined, the default `CacheIndexer` is used. |

# 2 Configuring the TestCustomerData Property File

This chapter covers configuring the MCA Services file TestCustomerData.properties.

## Prerequisites for Configuring TestCustomerData.properties

Before modifying any TestCustomerData.properties setting the file first needs to be extracted from the EAR file. Once the modifications have been made the property file is then added back into the EAR file, maintaining the original path.

***To configure TestCustomerData.properties***

1. Extract TestCustomerData.properties from the EAR file and open it in a text editor.
2. Edit the properties that need to be configured, following the instructions in this document.
3. Re-add the modified TestCustomerData.properties to the EAR file, maintaining the siebel.ear/resources path.

## Configuring this.absolutePath

The this.absolutePath property must be configured to point to the location of the TestCustomerData.properties file. The transactionHandler.test.customerData property in BankframeResource.properties should be set to the same value as this.absolutePath; this is covered in the next chapter.

***To configure this.absolutepath***

1. Open TestCustomerData.properties in a text editor. The following setting appears on the last line:

   this.absolutePath=/<hostname>/…/siebel.ear/resources/TestCustomerData.properties

2. Set this.absolutepath to the absolute path of the TestCustomerData.properties file. Adjust the path to your application server configuration.
   - Use the URL, IP address or machine name (without domain suffixes) of the installation machine if the EAR file is deployed on a remote machine.
   - Use the installation root if the EAR file is deployed on the local machine.

# 3 Configuring the BankframeResource Property File

This chapter covers configuring the MCA Services file BankframeResource.properties.

## Prerequisites for Configuring BankframeResource.properties

Before modifying any `BankframeResource.properties` setting the file first needs to be extracted from the EAR file. Once the modifications have been made the property file is then added back into the EAR file, maintaining the original path.

***To configure BankframeResource.properties***

1. Extract `BankframeResource.properties` from the EAR file and open it in a text editor.
2. Edit the properties that need to be configured, following the instructions in this document.
3. Re-add the modified `BankframeResource.properties` to the EAR file, maintaining the `siebel.ear/resources` path.

## Refresh Settings

### resource.cache.refreshInterval

This setting specifies how frequently the `BankframeResource.properties` file should be read in order to detect changes made to the file. The value is specified in seconds, and the default value is `900` seconds or 15 minutes.

## EJB Settings

### ejb.server

This setting specifies the URL of the EJB server to use. Possible values are:

`t3://localhost:7001` (WebLogic)

`iiop://localhost` (WebSphere)

## ejb.compliance

This setting specifies the EJB Specification level that the application complies with. Possible values are:

- `1.0` for Application servers that implement EJB Spec 1.0 (e.g. WebLogic 4 and earlier versions, WebSphere 3.5 and earlier versions)
- `1.1` for Application servers that implement EJB Spec 1.1

## ejb.initialContextFactory

This setting specifies the JNDI Initial Context Factory to use for creating JNDI Initial Contexts. Possible values are:

- `weblogic.jndi.T3InitialContextFactory` (WebLogic)
- `com.ibm.ejs.ns.jndi.CNInitialContextFactory` (WebSphere 3.5)
- `com.ibm.websphere.naming.WsnInitialContextFactory` (WebSphere 4.0 and later versions)

## ejb.jndiSyntax

This setting specifies the JNDI syntax to use for looking up EJB JNDI names. Possible values are:

- `1.0` for application servers that implement EJB Spec 1.0 JNDI Naming Conventions (e.g. WebLogic 4 and earlier versions, WebSphere 3.5 and earlier versions)
- `1.1` for application servers that implement EJB Spec 1.1 JNDI Naming Conventions

## ejb.jndiPrefix

This setting specifies the string to be prefixed to JNDI names when the `ejb.jndiSyntax` value is `1.1`. The prefix can be the start of any valid JNDI namespace, typical values include:

- `java:comp/env/` (For EJBs placed in the root JNDI environment context)
- `java:comp/env/ejb/` (For EJBs placed in the EJB environment context, this is recommended by the specification)

# RequestRouter settings

## requestRouter.entityJndiName

This setting specifies the JNDI Name of the entity EJB used to store route information, typical values include:

- `eontec.bankframe.Route` (DBMS based route information)

### requestRouter.jndiName

This setting specifies the JNDI name of the `RequestRouter` EJB, the default value is:

`eontec.bankframe.RequestRouter`

### requestRouter.addResponseStats

This setting determines if a response statistics `DataPacket` is added to the return `Vector` or not

Values are:

`True` – add response statistics `DataPacket` to return `Vector`

`False` – do not add response statistics `DataPacket` to return `Vector`

# Channel Management settings

### channel.client

This setting specifies the name of the class to use for sending client requests. This class must be a sub-class of `com.bankframe.ei.channel.client.ChannelClient`. The default value is:

`com.bankframe.ei.channel.client.HttpClient`

To use SSL set the value as follows: channel.client=com.bankframe.ei.channel.client.HttpsClient

### channel.enforcesingleton

This Boolean setting specifies the object creation behaviour of the `ChannelClientFactory`. When set to true, the factory will always return the same instance of the relevant `ChannelClient` object. When the `channel.enforcesingleton` property is set to false, the factory will create a new instance of the `ChannelClient` everytime. If `channel.enforcesingleton` is not specified, the factory will treat the default value as true.

### HttpClient settings

These settings are used to configure the `com.bankframe.ei.channel.client.HttpClient` class.

#### channel.http.client.url

This setting specifies the URL of the servlet for `HttpClient` to communicate with. Possible values are:

`http://localhost:7001/HttpServer` (e.g. WebLogic 5.1)

`https://localhost:7001/HttpServer` (e.g. WebLogic 5.1 for SSL)

`http://localhost/HttpServer` (e.g. WebSphere 3.5)

`https://localhost/HttpServer` (e.g. WebSphere 3.5 for SSL)

`http://localhost:7001/BankFrameMCA/HttpServer` (e.g. WebLogic 6.1 and later)

`https://localhost:7001/BankFrameMCA/HttpServer` (e.g. WebLogic 6.1 and later for SSL)

`http://localhost/BankFrameMCA/HttpServer` (e.g. WebSphere 4.0 and later)

`https://localhost/BankFrameMCA/HttpServer` (e.g. WebSphere 4.0 and later for SSL)

### channel.http.client.contentType

This setting specifies the MIME type to encode the client request as, the default value is:

```
application/x-eontec-datapacket-xml
```

### channel.http.client.header

This setting is used to define a properties subset of keys from the first DataPacket in the request, to be set as request properties in the connection. A key in the subset should match a key name in the DataPacket.. This setting is not mandatory.

### channel.http.client.addHeaderFieldsToSingleHttpHeaderField

This Boolean setting is used to specify how the request properties are set, either as one, delimited header field, or a series of header fields, each one corresponding to a channel.http.client.header key. This setting is mandatory only if a channel.http.client.header property subset is defined.

### channel.http.client.singleHeaderFieldName

This setting defines the header field name value to use if the channel.http.client.addHeaderFieldsToSingleHttpHeaderField propery value is true. This setting is mandatory only if the channel.http.client.addHeaderFieldsToSingleHttpHeaderField property value is true.

### channel.http.client.singleHeaderField.seperator

This setting defines the separator to use when header fields are being set in the request properties through a single header field. This setting is mandatory only if the channel.http.client.addHeaderFieldsToSingleHttpHeaderField propery value is true.

## HTTPSClient Settings

These settings are used to configure the `com.bankframe.ei.channel.client.HttpsClient` class.

### Channel.type.truststore

This setting determines the location of the trust store to be used by the HTTPS client, e.g.

```
channel.https.truststore=d:JDK1.4.1_02\lib\security\jssecacerts\truststore.jks
```

### channel.https.keystore

This setting determines the location of the keystore to be used by the HTTPS client, e.g.

```
channel.https.keystore=d:JDK1.4.1_02\lib\security\jssecacerts\certs.jks
```

### channel.https.keystorePassword

This setting determines the password to be used to access the keystore, e.g.

```
channel.https.keystorePassword=keypass123
```

### channel.https.ssl.protocol

This setting determines the SSL protocol class that is used in the `java.protocol.handler.pkgs` system property, for example,

`channel.https.ssl.protocol=com.sun.net.ssl.internal.www.protocol` (for WebLogic).
`channel.https.ssl.protocol=com.ibm.net.ssl.internal.www.protocol` (for WebSphere).

### channel.https.ssl.provider

This setting determines the JSSE (Java Secure Socket Extension) SSL provider class that is used in conjunction with channel.https.ssl.protocol, for example,

`channel.https.ssl.provider=com.sun.net.ssl.internal.ssl.Provider` (for WebLogic).

`channel.https.ssl.provider=com.ibm.jsse.JSSEProvider` (for WebSphere).

## Codec to MIME type mappings

Each client request MIME Type is handled by a specific codec class. The mapping has the following form:

`channel.http.codec.mapping.<mime-type>=<CodecClassName>`

For example:

`channel.http.codec.mapping.application/x-eontec-datapacket-xml=com.bankframe.ei.channel.codec.DPTPCodec`

`channel.http.codec.mapping.application/x-eontec-datapacket-hex=com.bankframe.ei.channel.codec.JOTPCodec`

## channel.codec.paddingstring

The DPTPPaddingCodec requires a padding string to be specified if that codec is to be used. The padding string is defined using channel.codec.paddingstring property. If not defined, the DPTPPaddingCodec will use the ^ character by default. The padding string is used to wrap the special characters < and > that are used by DPTPCodec.

# Client Connectivity Backwards Compatability

The following settings are retained for backwards compatibility with the previous Client Connectivity framework

## Backwards Compatibility - HTTP_SERVER

This setting specifies the URL of the HTTP Server. Possible values are:

`http://localhost:7001/` (WebLogic)

`http://localhost/` (WebSphere)

## Backwards Compatibility - SERVLET

This setting specifies the path to `BankframeServlet` on the HTTP Server. Possible values are:

`BankframeServlet` (e.g. WebLogic 5.1, WebSphere 3.5)

`BankFrameMCA/BankframeServlet` (e.g. WebLogic 6.1, WebSphere 4.0 and later)

## Backwards Compatibility - Servlet logging file

This setting is retained for backwards compatibility only & should not be used otherwise. It is not used by MCA Servlets. The default value is:

`servlets.log`

# Security settings

## security.provider

This setting specifies the Security Provider class that MCA should use. The specified class must implement the `com.bankframe.services.security.BankFrameSecurityProvider` interface. Possible values are:

`com.bankframe.services.security.NullBankFrameSecurityProvider` (Disable security)

`com.bankframe.services.security.DefaultBankFrameSecurityProvider` (Use default MCA security framework)

## security.sessionMgmtJndiName

This setting specifies the JNDI name of the Session Management EJB to be used by the `DefaultBankFrameSecurityProvider`. The default value is:

`eontec.bankframe.EJBSessionManagement`

## security.accessControlJndiName

This setting specifies the name of the Access Control EJB to be used by the `DefaultBankFrameSecurityProvider`. The default value is:

`eontec.bankframe.EJBAccessControl`

### security.sessionMgmt.defaultUserTimeoutPeriod

This setting specifies how long user sessions can be inactive before they are timed out. The value is specified in seconds. The default value is `900` seconds which is fifteen minutes.

# Logging Settings

### wl61.debugLoggingEnabled

This WebLogic-specific setting determines whether DEBUG level log messages should be forwarded to the WebLogic logging framework. This setting is case sensitive. The syntax is `wl61.debugLoggingEnabled=<true|false>`.

### wl61.redirectDebugToInfo

This WebLogic-specific setting determines whether DEBUG level log messages should be forwarded as INFO level messages to the WebLogic logging framework. This setting is case sensitive. The syntax is `wl61.redirectDebugToInfo=<true|false>`.

### log4j.config.path

This setting determines which LOG4J configuration file to use for configuring LOG4J. This setting must specify the absolute path to the properties file. The syntax is `log4j.config.path=</path/to/some/log4j.properties>`.

### log4j.config.refresh

This sets how often LOG4J checks its configuration file to see if any configuration changes have occurred. This value is specified in seconds. The syntax is `log4j.config.refresh=<time value in seconds>`.

### Console.logger

This setting specifies whether or not console logger debug messages are output. Configuring as `console.logger=DEBUG` enables debug level console logging. Leaving the `console.logger` value blank or not including the `console.logger` key in the `BankframeResource.properties` file disables console logging.

# Audit settings

## audit.provider

This setting specifies the class that MCA should use for auditing Financial Components. Possible values are:

`com.bankframe.services.audit.NullBankFrameAuditProvider` (Disable all auditing)

`com.bankframe.services.audit.DefaultBankFrameAuditProvider` (Enable default MCA Audit service)

# Localization settings

## localization.messageFile

This setting specifies the name of the resource bundle to use for localizable messages. The default value is:

`BankframeMessages`

# E-mail settings

## mail.smtpServer

This value specifies the name DNS name or IP address of the SMTP mail to use for sending e-mails from the MCA Mail Service. The default value is:

`mail.smtpServer=localhost`

# LDAP Settings

The LDAP Configuration settings are divided into two parts: default settings and settings specific to an LDAP context.

## ldap.default.java.naming.provider.url

This setting specifies the URL of the LDAP server. The default value is:

`ldap://localhost:389`

## ldap.default.java.naming.factory.initial

This setting specifies the JNDI initial context factory to use for resolving LDAP resources. The default value is:

```
com.sun.jndi.ldap.LdapCtxFactory
```

## ldap.default.java.naming.security.authentication

This setting specifies the authentication mechanism to use for connecting to the LDAP server. The default value is:

```
simple
```

## ldap.default.java.naming.security.principal

This setting specifies the user principal to use for connecting to the LDAP server. The default value is:

```
cn=bankframe,dc=eontec,dc=com
```

## ldap.default.java.naming.security.credentials

This setting specifies the user password to use for connection to the LDAP server. The default value is:

```
bankframe
```

## LDAP User Authentication settings

### bankframeusers.ldap.baseDn

This setting specifies the root LDAP context where User information is stored. The default value is:

```
ou=users,ou=accessgroups,o=bankframeMCA,dc=eontec,dc=com
```

### bankframeusers.ldap.defaultSearchFilter

This setting specifies the LDAP search filter used to locate users. The default value is:

```
uid={0}
```

Consult the JNDI Tutorial for more information on LDAP search filters.

## LDAP Session Management settings

### bankframesessions.ldap.baseDn

This setting specifies the root LDAP context where Session information is stored. The default value is:

```
ou=sessions,o=bankframeMCA,dc=eontec,dc=com
```

## LDAP Access Control and Routing settings

### bankframeroutes.ldap.baseDn

This setting specifies the root LDAP context where Route information is stored. The default value is:

`ou=routes,o=bankframeMCA,dc=eontec,dc=com`

### bankframeroutes.ldap.rdnAttribute

This setting specifies the primary key attribute of the route information. The default value is:

`eontecServiceId`

## LDAP Context Cache Setting

### Ldap.context.cache

This setting specifies whether or not server context caching is enabled, for example,

`ldap.context.cache=true` enables server context caching. Setting this to false disables server context caching.

# XML Settings

## xml.eDocBuilder.dtdLocation

This setting specifies the URL of the Eontec XML DTD. Possible values are:

`http://localhost:7001/bankframe/dtd/BankFrameProcess.dtd` (WebLogic 5 & earlier versions)

`http://localhost/bankframe/dtd/BankFrameProcess.dtd` (WebSphere 3.5 & earlier versions)

`http://localhost:7001/BankFrameMCA/dtd/BankFrameProcess.dtd` (WebLogic 6.1 and later)

`http://localhost/BankFrameMCA/dtd/BankFrameProcess.dtd` (WebSphere 4.0 and later)

## xml.eDocBuilder.systemId

This setting specifies a default location for DTD files of incoming XML Documents. (This is used as a back-up if the DTD is not specified with a full URL in incoming XML Documents)

So, if an incoming XML doc specifies its DTD with a line: `SYSTEM "BankFrameProcess.dtd"`, the parser will look for this file at the location specified by the `systemId` property. If the incoming XML doc specifies its DTD with a line `SYSTEM http://www.eontec.com/xml/dtd/BankFrameProcess.dtd` then the `systemId` property is ignored. The default value is:

`http://localhost/bankframe/dtd/`

## xml.parser.validating

This setting specifies whether the underlying XML parser used should be validating or non-validating. Possible values:

`true` (The input XML is checked for compliance with a DTD)

`false` (The input XML is not checked for compliance with a DTD)

## xml.parser.ignoreComments

This setting specifies whether the underlying XML parser should ignore comments or not. Possible values:

`true` (The parser ignores comment blocks)

`false` (The parser does not ignore comment blocks)

## xml.parser.ignoreElementContentWhiteSpace

This setting specifies whether the underlying XML parser should ignore white space or not. Possible values:

`true` (The parser ignores white space)

`false` (The input does not ignore white space)

## xml.parser.nameSpaceAware

Specify whether the underlying XML parser is namespace aware or not. Possible values:

`true` (The parser ignores white space)

`false` (The input does not ignore white space)

## xml.transformer.StyleSheetDir

Specify the URI location where transformation XSL style sheets are stored on the application server. The default value is:

`http://localhost/bankframe/stylesheets/`

## XSL Properties

For each XML request/response that is processed by applying an XSL transformation a mapping must be defined to associate the MIME `content-type` of the request/response with the appropriate XSL style-sheet to apply. For example:

```
channel.http.xml.xsl.request.content-type.application/
x-foo-request-xml=http://localhost/eontec/mca/stylesheets
/foo-xml-request.xsl

channel.http.xml.xsl.response.content-type.application/
x-foo-response-xml=http://localhost/eontec/mca/stylesheets
/foo-xml-response.xsl
```

The settings above specify that for requests of type: `application/x-foo-request-xml` the style-sheet located at: `http://localhost/eontec/mca/stylesheets/foo-xml-request.xsl` should be applied to the incoming request.

Similarly for responses of type: `application/x-foo-response-xml` the style-sheet located at:

http://localhost/eontec/mca/stylesheets/foo-xml-response.xsl should be applied to the outgoing response

# Financial Process Integrator Settings

## transactionHandler.dataSource.alwaysCloseConnection

This setting specifies whether to close the DataSource when finished with it. Possible values are:

`true` (WebSphere)

`false` (WebLogic)

## transactionHandler.dataSource.jndiName

This setting specifies the JNDI name of the data source that the Financial Process Integrator should use. Default value is:

```
jdbc/bankfrm
```

## transactionHandler.dataSource.username

This setting specifies the username to use to connect to the database. If the App Server creates connections without using username and password then this line should be commented out. Default value is:

```
bankfrm
```

## transactionHandler.dataSource.password

This setting specifies the password to use to connect to the database. If the App Server creates connections without using username and password then this line should be commented out. Default value is:

`bankfrm`

### transactionHandler.test.customerData

This setting specifies the location of the file used by the `TestCustomer` sample Host Connector. Possible values are as follows – adjust the path according to the location of the file on your system. A single back-slash cannot be used on Windows as it needs to be escaped.

`/opt/WebSphere/AppServer/installedApps/server1/siebel.ear/TestCustomerData.properties` (e.g. WebSphere 5.0 on UNIX)

`\\WebSphere\\AppServer\\installedApps\\server1\\siebel.ear\\TestCustomerData.properties` (e.g. WebSphere 5.0 on XP)

`/opt/bea/user_project/mydomain/TestCustomerData.properties` (e.g. WebLogic 8.1 and later on UNIX)

`\\bea\\user_project\\mydomain\\TestCustomerData.properties` (e.g. WebLogic 8.1 and later on XP)

Note: The WebLogic path to the TestCustomerData.properties file must not include the EAR as the file will not be found. Therefore, extract the TestCustomerData.properties file and place it in the server's domain folder ensuring the setting in the BankframeResource.properties file points to this location. The exception to this is if the EAR is on the server as an exploded EAR, in which case the TestCustomerData.properties file will be found.

## BMP EJB Persister Settings

### persister.default

This setting defines a default persister to use for all BMP EJBs

Alternatively each BMP EJB can explicitly specify which persister it uses by defining a key of the form: `persister.jndiName`, where `jndiName` is the JNDI name of the EJB

The default value is:

`com.bankframe.ei.txnhandler.persister.TxnPersister`

### persister.cache.updateOnAmend

This setting determines if BMP EJB `amend()` calls update the cache or remove the cache entries. Possible values:

`yes` (The cache is updated)

`no` (The cache is not updated, the relevant data in the cache is removed so it can be re-read from the host)

## EJB Specific Persister Settings

### Customer EJB

This setting specifies the Persister class to use for the example Customer EJB. The default value is:

```
com.bankframe.ei.txnhandler.persister.MasterEntityPersister
```

### Address EJB

This setting specifies the Persister class to use for the example Address EJB. The default value is:

```
com.bankframe.ei.txnhandler.persister.TxnPersister
```

### Account EJB

This setting specifies the Persister class to use for the example Account EJB. The default value is:

```
com.bankframe.ei.txnhandler.persister.TxnPersister
```

## session.amend.persister.default

The Session Amend Helper service is deprecated and has been replaced by Transaction Handler Broker.

This setting defines a default persister to use for all BMP EJBs

Alternatively each BMP EJB can explicitly specify which persister it uses by defining a key of the form: `session.amend.persister.jndiName`, where `jndiName` is the JNDI name of the EJB

The default value is:

```
com.bankframe.ei.txnhandler.sessionamendpersister.TxnSessionAmendPersiste
r
```

# MCA Host Connector settings

These settings specify which MCA Host Connectors should be deployed. The settings use the following pattern:

```
transactionHandler.connector.<connector-name>.ObjectFactory_Impl=<class-
name>

transactionHandler.connector. <connector-
name>.ConnectionFactory_Impl=<class-name>

transactionHandler.connector. <connector-
name>.ManagedConnectionFactory_Impl=<class-name>

transactionHandler.connector. <connector-name>.maxConnections=<max-pool-
size>

transactionHandler.connector. <connector-name>.timeOut=<inactivity-
timeout>
```

where:

`<connector-name>` is the name of the connector

`<class-name>` is the name of a java class

`<max-pool-size>` is the maximum size of the connection pool

`<inactivity-timeout>` is the period of time in milliseconds to wait before removing inactive connections

# Financial Process Integrator Store for Forward Settings

## transactionHandler.storeAndForward.forwardingDelay

This setting is used by the default constructor of the `ForwardingThread` to set the time interval, in milliseconds, between batches being sent to the host.

```
transactionHandler.storeAndForward.forwardingDelay=2000
```

## transactionHandler.storeAndForward.hostStatusDelay

This setting is used by the default constructor of the `HostStatusMonitor` to set the time interval, in milliseconds, to wait between checks on the host status.

```
transactionHandler.storeAndForward.hostStatusDelay=30000
```

## transactionHandler.storeAndForward.url

This setting is used to specify the url of the `ForwardTransactionServlet`

```
transactionHandler.storeAndForward.url=http://localhost:7001/ForwardTrans
actionServlet
```

## transactionHandler.storeAndForward.startHostMonitorAutomatically

This setting is used to specify whether or not the `HostStatusMonitor` starts up automatically when the App server is started or not. It can have a setting of either `true` or `false`.

```
transactionHandler.storeAndForward.startHostMonitorAutomatically=true
```

## transactionHandler.storeAndForward.nextTransactionBatchAmount

This setting is used to specify the amount of transactions the `ForwardingThread` is to forward in a batch.

```
transactionHandler.storeAndForward.nextTransactionBatchAmount=50
```

# Caching Framework Settings

Below is a section of the `BankframeResource.properties` file showing the configuration for the cache for the `DESTINATION` table:

```
cache.destinationCache.class=com.bankframe.services.cache.GenericCache
cache.destinationCache.persister=com.bankframe.ei.com.bankframe.
ei.txnhandler.impl.destination.DestinationCachePersister
cache.destinationCache.policy=com.bankframe.services.cache.
LruCachePolicy
cache.destinationCache.policy.maxSize=100
cache.destinationCache.policy.thrashAmount=10
```

Note how the settings are named, they start with a prefix: `cache.`, followed by the name of the cache (in this case `destinationCache`) and then a suffix indicating the name of a specific configuration parameter (for example `.class`). Caches are created and obtained through CacheFactory. The CacheFactory will look up a subset of settings for the cache using `cache.<cache name>`. Subset keys will include class, to define which Cache class is to be used, this is mandatory. Subset keys may also include policy and persister keys.

Below is an explanation of each setting:

## cache.cleaninterval

This setting is used to specify in milliseconds how often any cache should ask its policy, if one is defined, to perform a clean up. This setting is used for all caches in the JVM. If not defined, the value will default to 10000 - i.e. 10 seconds. Note that this setting is intended for performance enhancement and the interval should ideally not be less than the lowest policy timeout value. This is to avoid situations where the policy has the overhead of checking for entries to remove, but none have timeout out since the last clean up.

## Cache settings

`class`: This is the fully qualified name of the cache class to use for this cache. This class must implement the `com.bankframe.services.cache.Cache` interface. If the cache requires a persister it must implement the `com.bankframe.services.cache.PersistentCache` interface.

`persister`: This is the fully qualified name of the persister class that should be used with this cache to retrieve data from the data store. This class must implement the `java.util.Map` interface. Some caches do not have a persistent store associated with them, so they will not need to specify a `persister` setting, in this case the `persister` setting should be omitted from the cache configuration settings. Note that this class is not related to the Financial Process Integrator concept of a persister.

`policy`: This is the fully qualified name of the cache policy class to use for this cache. This class must implement the `com.bankframe.services.cache.ConfigurableCachePolicy` interface.

## Policy Specific Settings

Each policy object can have its own settings that configure how it behaves. The settings for each of the policy objects provided with MCA are detailed below:

### LruCachePolicy

This policy uses a least recently used algorithm to limit the cache to a specified maximum size. This policy has the following configurable settings:

- `maxSize`: This specifies the maximum number of entries permitted in the cache. When this is exceeded the least recently used entries are removed from the cache until the cache size is reduced to the maximum size.
- `thrashAmount`: When the maximum size of the cache is exceeded this policy tries to remove just enough entries to reduce the cache to the maximum size. This setting can be used to force the policy to reduce the number of cache entries to `maxSize` less `thrashAmount`. This means that when the cache size is exceeded the least recently used entries are removed so that space is left for new entries to be added.

### TimeoutCachePolicy

This policy removes entries that have not been used for more than a specified period of time. This policy has the following configurable setting:

- `timeout`: This value indicates the maximum time in seconds that an entry can remain in the cache without being used.

### PerEntryTimeoutCachePolicy

This is a new policy object introduced in this release. This policy is similar to the `TimeoutCachePolicy` except that each individual entry in the cache can have its own timeout setting. This timeout value needs to be specified programmatically for each entry in the cache by calling the `setTimeout(Object key, long timeout)` or `setTimeout(Set keys, long timeout)` methods of this class. Therefore this policy has no configurable settings

### cache.index.<INDEX_NAME>

This setting specifies the name of a cache index and the `CacheIndexer` class for managing the cache index. If there is no entry in the `BankframeResource.properties` file for `INDEX_NAME`, the `CacheIndexFactory` returns an instance of `CacheIndex` by default. If a specific `CacheIndexer` is not defined, the default `CacheIndexer` is used. The syntax is
`cache.index.<INDEX_NAME>=<CacheIndexerClass>`

# Session Affinity Settings

To configure Session Affinity the `BankframeResource.properties` file must be modified in two places.

- Firstly to notify the State Machine to include a unique token with every request the following must be set to true:
  `include.session.id=true`
- Secondly a configurable name must be specified as a key for the unique token when placed within a HTTP's request header. Therefore set the following:
  `channel.http.client.header.HTTP_HEADER_ID=SM_SESSION_ID`
  where `HTTP_HEADER_ID` is the configurable key.
  Note: the State Machine always places a key named `SM_SESSION_ID` in each request so this is not to be altered in the above setting.

# Request Context Settings

To configure Request Contexts the `BankframeResource.properties` file must be modified as follows.

Specify a RequestContextFactory like below
requestContext.factory=com.bankframe.services.requestcontext.*PreferredRequestContextFactory*
where PreferredRequestContextFactory is used to create and associate state with the preferred RequestContext.

Note: If this setting is not modified the default `NullRequestContextFactory` will be used which doesn't associate any context with a request.

# Entitlements Settings

To configure Entitlements, the `BankframeResource.properties` file must be modified to point to the `TaskChannelEntitlementsSQLUtilities` class as follows.

entitlements.sql.utilities=com.bankframe.util.shared.accesscontrol.TaskChannelEntitlementsSQLUtilities

Note: this is set by default in `BankframeResource.properties`.

The caching must also be configured for Entitlements. Configuring caches is discussed in the Caching Framework section above. The default settings are already configured in `BankframeResource.properties`.

# JMS Caching Settings

To configure JMS Caching, the `BankframeResource.properties` file must be modified by doing the following steps:

## Node Identifier

Each node must be uniquely identified within a cluster to make sure that the node receiving a message knows the originator node, as shown in the following example.

jms.cluster.node.id=node1

## JNDI Context Factory

The Context Factory for doing JNDI lookups needs to be set up as follows:

For WebLogic:

jms.jndi.jndi.factory=weblogic.jndi.WLInitialContextFactory

For WebSphere:

jms.jndi.jndi.factory= com.ibm.websphere.naming.WsnInitialContextFactory

### JMS Connection Factory JNDI

Set the JNDI name of the JMS Connection Factory as follows:

jms.jndi.connection.factory=eontec.jms.TopicConnectionFactory

Note: This JNDI name must be the same as the JNDI name of the Connection Factory one specifies in the web console of the application server.

### JMS Topic JNDI

Set the JNDI name of the JMS Topic as follows:

jms.jndi.topic=eontec.jms.topic

Note: This JNDI name must be the same as the JNDI name of the Topic one specifies in the web console of the application server. In the case of a WebLogic installation, if the setting is changed from 'eontec.jms.topic' the `weblogic-ejb-jar.xml` of the message driven bean `com.bankframe.services.cache.JMSListener` must also change as the JNDI of the topic is also specified there.

## Timing Point Settings

### timingPoint.enabled

This setting determines whether or not timing points are enabled. Valid values are as follows:

`true` (use timing points)

`false` (Do not use timing points)

### timingPoint.writePointsToDisk

This setting determines whether or not timing points are written to the console. Valid values are as follows:

`true` (write timing points to console)

`false` (do not write timing points to console)

## timingPoint.subsystem.BANKFRAME.MCA

Set timingPoint.subsystem.BANKFRAME.MCA as follows

```
timingPoint.subsystem.BANKFRAME.MCA=BANKFRAME.MCA
```

## timingPoint.doSummary

This setting determines whether a summary of timing point information is displayed on the console or in a file. Valid values are as follows:

`true` (display summary )

`false` (Do not display summary

## timingPoint.fileName

This setting determines the name of the file to which timing point details are flushed.

Set the valueto timingPoint.fileName=timingpoints.log to write to the file timingpoints.log

## timingPoint.bufferSize

This setting determines the maximum size of the buffer to hold the timing points

The default setting is `timingPoint.bufferSize=1000`. Once this is exceeded all timing points will be flushed to file or console.

## timingPoint.analyzerClassName

It is possible to use different classes to analyse the timing point information

Set the value as follows:
timingPoint.analyzerClassName=com.bankframe.services.trace.NullTimingPointAnalyzer - where com.bankframe.services.trace.DefaultTimingPointAnalyzer is the name of the analyzer class to process timing points - to turn the analysis off

# MessageDigest.algorithm Setting

This setting determines the algorithm to use for encryption within MCA Services. Valid values are determined by the JCA – see your JCA documentation for further information. Typical values include `MD5` or `SHA-1`. Set to no value to disable encyrption.

The value is set as follows: `messageDigest.algorithm=SHA-1`

Refer to the MCA Services User Authentication documentation in the Enterprsie Services section for further information

# Remote Notification Settings

## TargetPort

This setting determines the port the client machine will use to listen for remote notification. The default value is: `targetPort=1196`

## SourcePort

This setting determines what port is used by the source of any notification messages. Any subsequent replies to that source will be routed to the specified port. The default setting is:

```
sourcePort=1198
```

## Timeout

This setting determines the timeout period in milliseconds for sending a remote notification. The default value is: `timeout=6000`

## Retries

This setting determines the number of retries the Notification module will make when posting a message. The default value is: `retries=3`

## ResponseLogFile

This setting determines the location of the log file where responses to remote notification are stored. The default value is:

```
responseLogFile=/export/home/bea/user_projects/eontec/response.log
```

## PayloadLogFile

This setting determines where to log the payload of any notification messages to - the default value is:

```
payloadLogFile=/export/home/bea/user_projects/eontec/payload.log
```

## TargetSelectionFactory

This setting determines the class to use to decide what target to use for a given source:

```
targetSelectionFactory=com.bankframe.services.notification.targetselectio
n.DefaultTargetSelectionFactoryImpl
```

### Rmi.remotePort

This setting determines what port the remote notification service listens on. The default value is:

```
rmi.remotePort=1099
```

### rmi.remoteNotificationURL

This setting determines the URL on which the remote notification service is available. The default setting is:

```
rmi.remoteNotificationURL=rmi://development.eon.ie:1099/RMINotificationSe
rvice
```

## Sequence Generation

This service allows for the replacement of the SystemControl table. This service utilizes the sequences within the database to cater for incremental values, e.g. DraftNumber, CustomerNumber. The settings include the following:

```
mca.services.sequences.factoryClass=com.bankframe.services.sequences.Orac
leSequenceGeneratorFactoryImpl
```

```
mca.services.sequences.datasource=bankfrm
```

## CRC Server Online Status

The `crc.system.online` Branch Teller CRC setting allows calls to the CRC server to be turned off when the server is not online. Setting `crc.system.online` to `true` indicates that the server is online and turns on calls to the CRC server

## Financial Process Integrator Broker Settings

The Financial Process Integrator Broker (TxnHandlerBroker) replaces the deprecated Session Amend Helper and is intended for use when an entity bean does not easily map to a host transaction. It provides a find and amend interface, but also factory classes for creating data sets for use with the Financial Process Integrator.

### transactionHandler.broker.hosttransactionfactory.<ejbname>

Setting used to define the HostTransactionFactory class for use with a given ejb. This is not a mandatory setting. If a setting can not be found for a particular ejb, then the default valu is used.

## transactionHandler.broker.hosttransactionfactory.default

This mandatory setting is used by TxnHandlerBroker to get default HostTransactionFactory class if none can be found for a given ejbname. Default value is com.bankframe.ei.txnhandler.broker.HostTransactionObjectFactoryImpl.

## transactionHandler.broker.persister.<ejbname>

Used by TxnHandlerBroker to get Persister class for the given ejbname

## transactionHandler.broker.persister.default

This mandatory setting is used by TxnHandlerBroker to get default Persister class if none can be found for a given ejbname. Default value is com.bankframe.ei.txnhandler.persister.TxnPersister

## transactionHandler.broker.removeFromCacheOperation.<ejbname>.<methodname>

Boolan setting used by TxnHandlerBroker to get the type of cache cleanup for the given ejbname and methodname

## transactionHandler.broker.removeFromCacheOperation.default

This mandatory Boolean setting is used by TxnHandlerBroker to get the default type of cache cleanup if none can be found for a given ejbname. Default value is true (i.e. values in cache are removed)

# 4 Administrating MCA Services

MCA Services provides ServiceServlet, which is an administration tool that simplifies managing the following:

- Routes
- Sessions
- Users and Groups
- Sending DataPackets to Financial Components

To use this administration tool, type the following URL into a browser:

http://<hostname>:<portnumber>/BankFrameMCA/ServiceServlet

## Configuring MCA Routing

Every time a new route or service is developed it must be added to the list of routes. This list of routes is stored in the ROUTES table in the database. However, if a privileged User (such as a System Administrator) wishes to delete or amend the properties of a particular route then the MCA Route Configuration tool can be used. The configuration tool is used instead of running SQL on the services table every time the attributes of a route need to be changed.

### Initializing the Route Configuration Tool

From the admin tool screen click the Configure Route link.

### Creating a New Route

Select “create a new route”

Input the details for the new route:

- REQUEST_ID.
- JNDI Name
- Description.

Also, highlight whether or not this bean should be subject to session management. If it is, then every request on this route will be checked for a sessionId before the request is fulfilled. If it is not session managed then the bean is said to be a “free route” (i.e. a session does not have to be established to avail of the bean’s services). For more information on session management please consult the Session Management documentation.

## Other functionality provided by the route configuration tool

- List all existing routes.
- Search for a particular route.
- Delete a route.

# Administrating MCA Sessions

As part of the session management facility offered by MCA, there is a HTML based configuration tool provided for managing each currently active or expired session.

Note that this section assumes knowledge of the session management model used by MCA Services, refer to the MCA Services Developer Guide.

To select the Administrating MCA Sessions operations click the Administer Sessions link.

## Listing All Current Sessions

Selecting the List All Current Sessions link displays all registered MCA sessions. For each live session the following are displayed:

- The session ID
- The user ID associated with the session
- The last time this user accessed the system

## Removing Expired sessions

Selecting the Remove Expired Sessions link removes all expired sessions. A message displays stating how many expired sessions were removed.

## Removing All Sessions

Selecting the Remove All Sessions link removes all sessions, both current and expired. A message displays stating how many sessions were removed.

## Deleting a Specific Session

Selecting the Delete a Specific Session link removes the specified session. Enter the session ID of the session to be deleted. A message displays stating whether or not the session was deleted successfully.

# Administrating MCA Users and Groups

MCA users, groups, and the permissions associated with them are held in several database tables. MCA provides an HTML based configuration tool to maintain these tables.

Note that this section assumes knowledge of the user/group model used by MCA; refer to the Access Control documentation in the MCA Services Developer Guide.

To access the MCA User and Groups administration operations click the Administer Users and Groups link.

## Administrating MCA Users

### View all Users

This will allow you to view all registered users with MCA.

### Create a new User

This allows you to create a new User. To do this, you must supply the following:

- User ID
- User Name
- Password

### Delete a User

This will delete a user and all other details associated with that user from the system.

### View a User's Permissions

This command will show all the permissions assigned to a user. It displays both the permissions a user may have outside the scope of his/her group(s) and also the group(s) which s/he has membership of.

### Assign/Extend a user's permissions

A user can be given extra permissions outside the scope of his/her assigned group(s). The Assign/Extend user permissions command will display a table showing the permissions a user does not have. The permissions a user has are an amalgamation of the permissions his/her group(s) has been assigned, plus any permissions s/he has been assigned outside the scope of his/her group(s).

Tick the checkbox beside any permission you want to assign and then hit the 'UPDATE USER' button on the screen to update the user with this permission.

### Remove a user's permissions

This command will remove a user's permissions. It will show a table of a user's assigned permissions outside the scope of his/her assigned group(s). No permissions relating to group membership will be displayed. To remove a permission from a user, tick the box of the permission(s) you wish to remove from the user and hit the 'REMOVE USER PERMISSIONS' button.

### Assign user to group(s)

If you choose this command, you will be prompted for a user ID. After this you will be given a list of the groups that this user is not a member of. To make the user a member of one or more of the groups, tick the checkbox of the group(s) and hit the 'ASSIGN USER TO GROUP' button.

### Remove user from group(s)

Once you have input an appropriate user ID, you will be given a screen showing all the groups this user is a member of. To remove the user from one or more groups, tick each group's checkbox and hit the 'DELETE FROM GROUP' button.

## Administrating MCA Groups

### View all groups

This command will allow you to view all the groups registered with MCA.

### Create a new Group

This command will allow you to create a new group. To do this you must supply the following:

- Group ID
- Group Name

### View a group's members

This command will allow you to see all users registered with a particular group. It will initially prompt for a group ID.

### Delete a group

This command will delete a group and all other entries associated with it. For example, if you delete group 'Employee' and user 'X' is in this group, all information relating user 'X' to group 'Employee' will be removed from the system.

### View a group's permissions

This command will allow you to view the permissions associated with a group. It will prompt for a group ID.

### Assign/Extend a group's permissions

This command will allow you to add permissions to a given group. To add a permission to a group, tick the checkbox of the permission(s) you want to add and hit the 'UPDATE GROUP' button.

### Remove a group's permissions

This will allow you to remove the permissions of a given group. To remove the permission(s) of a group, tick the checkbox of the group and hit the 'REMOVE GROUP PERMISSIONS' button.

# Using the MCA Monitor Utility

The MCA architecture processes `DataPackets` from various channels and routes them to the correct Financial Component based on a `REQUEST_ID` specified in the `DataPacket`. Although these `DataPackets` will be generated by various client types, MCA provides a HTML based utility called "Monitor", which explicitly allows a user to generate `DataPackets` of varying formats, forward them to their correct Financial Component, and then receive the response `DataPacket`(s) from the Financial Component.

Please note that this section assumes knowledge of the `DataPacket` data structure and the concept of a `REQUEST_ID` as it applies to MCA.

## Initializing and Using the Monitor Servlet

Click on the “BankFrame MCA Monitor” link to launch the monitor servlet

The Monitor utility will be explained using a logon request DataPacket example. A DataPacket for a logon will have the following structure and example data:

| DATA PACKET NAME | LOGON REQUEST |
|---|---|
| OWNER | Siebel |
| REQUEST_ID | 50000 |
| userPassword | AAAAA |
| USER_ID | Kds |
| LOGON | true |

Key in the appropriate values for the default text fields supplied, namely DATA PACKET NAME (LOGON REQUEST), OWNER (Siebel) and REQUEST_ID (50000). Ensure that the value you use for REQUEST_ID is appropriate. Although 50000 is the REQUEST_ID used here, this may not be the correct REQUEST_ID for the User Authentication process in your own environment.

When you have done this, key in the values for the extra fields you require. This can be done via the “Add a new field” command. Using this, key in the values for userPassword (AAAAA), USER_ID (Kds), LOGON (true). Fields can be added to and removed from a DataPacket by using the “Add a new Field” & “Remove a Field” commands, respectively.

Also, it is important to note that the “Current DataPacket” field on the HTML page, and *not* the information in the text fields, represents the information sent to the server. If the information in the text fields is not the same as the information in this field, then hit the “Update” button on this page to update the “Current DataPacket” field.

Following this, hit “Send DataPacket” and you will receive the response DataPacket for this Financial Component.