Softlink

Web Service API

Liberty and Oliver v5

Softlink

www.softlinkint.com

Overview

This document outlines the Web Services API available in both Liberty and Oliver v5. It is intended to be read by system integration developers familiar with Web Services in any application language.

The Web Services offering has four main components:

- Generic entity API (since v4.2 build 7)
- OPAC specific API (since v4.2 build 6.0 RC 1)
- Federated Search specific API (since v5 build 1)
- An implementation of the Open Search API (since v4.2 build 6.0 patch 9)

The generic entity API allows you to perform basic create, read, update and delete operations on any system entity.

The OPAC specific API allows you to display borrower details, search catalogue records and renew a loan.

The Federated Search specific API allows third-party federated search products to search catalogue records.

The Open Search module implements version 1.1 of the Open Search specification. This supports OPAC searching from the browser search box and federated searching with *MS SearchServer 2008*. A federated search is the simultaneous querying of multiple online databases (locations) for the purpose of generating a single search results page for end users.

Web Services Setup and Configuration

This section describes setup and configuration that is either required prior to using the web services or to control their behaviour.

A registration module is required to access Web Services features. There are four available registration options:

- OpenSearch only
- Federated Search API and OpenSearch
- OPAC Web Services API (includes Federated Search API) and OpenSearch
- All APIs – Generic, OPAC, Federated Search and OpenSearch

The built-in “Web Services” client account is intended to be used when accessing the Web Services API. The account is pre-configured with the “Web Services” user role which is linked to a set of user privileges allowing the availability of API operations to be controlled.

When deploying web services, you should modify the privilege settings to only allow operations you intend to use, for example allowing access to browse any system entity, but only allowing the Client entity to be updated.

***System Parameters**:*

webServicesEncryptCredentials – Parameter number 1390. When enabled assumes the Web Services alias and password parameters are RSA encrypted. In v4.2 build 7 onwards, this parameter is controlled by Softlink Support.

OPAC Search Web Services Functions

These functions were first introduced in v4.2 build 6.0 RC 1.

WSDL location:

http://<server>:<port>/<application-prefix>/OpacAccess?wsdl

Note you will need access to the application log file in order to debug your Web Services client implementation – access errors are not usually reported in an API response.

1. GetBorrowerDetails

Returns the details (personal details, current loans, loan history, current bookings and current reservations) about a specified borrower.

This function requires the Web Services username/password on this system as an added layer of security.

***Arguments**:*

corpAlias: The alias of the corporation on this system we'll load from.

clientAliasEnc: The alias of the Web Services user on this system whose login should be used. Should be RSA encrypted depending on parameter 1390.

clientPasswordEnc: The password of clientAlias. Should be RSA encrypted depending on parameter 1390.

borrowerAliasEnc: The alias of the borrower on this system whose details are to be returned. Should be RSA encrypted depending on parameter 1390.

Returns the borrower's details in XML format.

Throws an IllegalArgumentException if no user matching clientAliasEnc/clientPasswordEnc or borrowerAliasEnc could be found (or there was a problem decrypting).

***Example request**:*

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:opac="http://opac.webservice.libraryserver.softlink.net/">
   <soapenv:Header/>
   <soapenv:Body>
      <opac:getBorrowerDetails>
         <corpAlias>MyLibrary</corpAlias>
         <clientAliasEnc>nnn</clientAliasEnc>
         <clientPasswordEnc>nnn</clientPasswordEnc>
         <borrowerAliasEnc>nnn</borrowerAliasEnc>
      </opac:getBorrowerDetails>
   </soapenv:Body>
</soapenv:Envelope>
```

Example reply:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:getBorrowerDetailsResponse
xmlns:ns2="http://opac.webservice.libraryserver.softlink.net/">
<return><![CDATA[<net.softlink.libraryserver.webservice.opac.OpacResult>  <valueObjects
class="list">
    <Client>
      <uuid>f283add3c0a814300187ceb3a7fb98f6</uuid>
      <updateSearchIndex>true</updateSearchIndex>
      <corpId>0</corpId>
      <userRoleUuid>f281ba41c0a81430011d9bfffe308550</userRoleUuid>
      <mailTitle></mailTitle>
      <gender></gender>
      <expiryDate class="sql-timestamp">2009-07-19 00:00:00.0</expiryDate>
<borrowerLoanCategoryUuid>c631560475824edab42603c66784cca5</borrowerLoanCategoryUuid>
      <defaultOperator></defaultOperator>
      <deposit>0.00</deposit>
      <personalName></personalName>
      <familyName></familyName>
      <altContact></altContact>
      <borrowerTypeUuid>4530b7f1c0a8142801617189ad6b9122</borrowerTypeUuid>
      <isARTEmailFormat>false</isARTEmailFormat>
      <memberSince class="sql-timestamp">2009-01-20 00:00:00.0</memberSince>
      <postCode></postCode>
      <mobilePhone></mobilePhone>
      <preferredMessageType></preferredMessageType>
      <sortableName></sortableName>
      <reservations class="list" />
      <loans class="list" />
      <loanHistorys class="list" />
    </Client>
  </valueObjects>
</net.softlink.libraryserver.webservice.opac.OpacResult>]]></return>
</ns2:getBorrowerDetailsResponse>
   </S:Body>
</S:Envelope>
```

2. Query

Performs the given query on this system and returns the search results.

This function requires the Web Services username/password on this system as an added layer of security.

***Arguments**:*

corpAlias: The alias of the corporation on this system we'll load from.

clientAliasEnc: The alias of the Web Services user on this system whose login should be used. Should be RSA encrypted depending on parameter 1390.

clientPasswordEnc: The password of clientAlias. Should be RSA encrypted depending on parameter 1390.

Search: The SLKQL search to perform.

maxResultCount: The maximum number of uuids to return.

Returns String Search results in XML format.

Throws IllegalArgumentException if no user matching clientAliasEnc/clientPasswordEnc could be found (or there was a problem decrypting), or if the search was an invalid syntax.

***Example request**:*

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:opac="http://opac.webservice.libraryserver.softlink.net/">
<soapenv:Header/>
   <soapenv:Body>
      <opac:query>
         <corpAlias>MyLibrary</corpAlias>
         <clientAliasEnc>nnn</clientAliasEnc>
         <clientPasswordEnc>nnn</clientPasswordEnc>
         <search>cats</search>
         <maxResultCount>1</maxResultCount>
      </opac:query>
   </soapenv:Body>
</soapenv:Envelope>
```

***Example reply**:*

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:queryResponse xmlns:ns2="http://opac.webservice.libraryserver.softlink.net/">
<return><![CDATA[<net.softlink.libraryserver.webservice.opac.OpacResult>  <valueObjects
class="list">
    <Catalog>
      <uuid>a77e9be3c0a8143d005b7127ef7c2427</uuid>
      <articleAbstract></articleAbstract>
      <articleText></articleText>
      <catalogId>104054</catalogId>
      <classification>636.800994</classification>
      <controlNo>2408556</controlNo>
      <description>139 p. : ill. ; 25 cm.</description>
      <edition>Rev. ed.</edition>
      <gmdUuid>e0edc847c0a81445018c1f36bd2ffb7d</gmdUuid>
      <isbn>0730102971 :</isbn>
      <normalIsbn>9780730102977</normalIsbn>
```

```
      <normalIsbn10>0730102971</normalIsbn10>
      <notes>First published 1983 by The Currawong Press with the title: Complete book of cats of
Australia. Includes
        index.</notes>
      <place>Frenchs Forest, N.S.W.</place>
      <publicationDate>1989</publicationDate>
      <publicationYear>1989</publicationYear>
      <responsibility>Barbara Walcott (Sepaki) &amp;amp; Dorothy Rickards.</responsibility>
      <sortableClassification>2.636.800994</sortableClassification>
      <sortableTitle>Complete book of cats in Australia</sortableTitle>
      <title>Complete book of cats in Australia</title>
      <bibliographicType>
        <uuid>4530bb5cc0a814280161718984fe5cd3</uuid>
        ...
      </bibliographicType>
      <catalogAttachments class="list" />
      <catalogAuthors class="list" >
        <CatalogAuthor>
          ...
          <author>
          ...
          </author>
        </CatalogAuthor>
      </catalogAuthors>
      <catalogCorporateAuthors class="list" />
      <catalogDatas class="list" />
      <catalogIsbns class="list" />
      <catalogPublishers class="list"/>
      <catalogRelations class="list" />
      <catalogTopicLists class="list" />
      <catalogSubjects class="list" />
      <catalogTitles class="list" />
      <catalogUrls class="list" />
      <gmd>
        ...
      </gmd>
      <issues class="list" />
    </Catalog>
  </valueObjects>
</net.softlink.libraryserver.webservice.opac.OpacResult>]]></return>
      </ns2:queryResponse>
   </S:Body>
</S:Envelope>
```

3. RenewLoan

Attempts to renew a specified item for a specified borrower and returns the success/failure of the renewal.

This function requires the Web Services username/password on this system as an added layer of security.

***Arguments**:*

corpAlias: The alias of the corporation on this system we'll load from.

clientAliasEnc: The alias of the Web Services user on this system whose login should be used. Should be RSA encrypted depending on parameter 1390.

clientPasswordEnc: The password of clientAlias. Should be RSA encrypted depending on parameter 1390.

borrowerAliasEnc: The alias of the borrower on this system whose details are to be returned. Should be RSA encrypted depending on parameter 1390.

itemBarcode: The barcode of the item to be renewed.

Return value is the success/failure of the renewal.

Throws IllegalArgumentException if no user matching clientAliasEnc/clientPasswordEnc or borrowerAliasEnc could be found (or there was a problem decrypting).

***Example request**:*

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:opac="http://opac.webservice.libraryserver.softlink.net/">
   <soapenv:Header/>
   <soapenv:Body>
      <opac:renewLoan>
         <corpAlias>SaharaDB2</corpAlias>
         <corpAlias>MyLibrary</corpAlias>
         <clientAliasEnc>nnn</clientAliasEnc>
         <clientPasswordEnc>nnn</clientPasswordEnc>
         <borrowerAliasEnc>nnn</borrowerAliasEnc>
         <itemBarcode>B557</itemBarcode>
      </opac:renewLoan>
   </soapenv:Body>
</soapenv:Envelope>
```

Generic Web Services Functions

The following API allows you to perform basic create, read, update and delete operations on any system entity. These functions were first introduced in v4.2 build 7.

***WSDL location**:*

http://<server>:<port>/<application-prefix>/EntityAccess?wsdl

1. Browse

Retrieve the contents of one or more database records.

***Arguments**:*

corpAlias : The alias of the corporation on this system we'll load from.

clientAliasEnc: The alias of the Web Services user on this system whose login should be used. Should be RSA encrypted depending on parameter 1390.

clientPasswordEnc: The password of clientAlias. Should be RSA encrypted depending on parameter 1390.

entityName: Indicates the type of entity to perform the operation on.

Uuids: List of one or more primary keys.

loadConstant: Specifies required deep loading. See l3_PageField.l3_loadConstant for examples.

Returns a list of value objects matching the given criteria in XML representation.

Example request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ent="http://entity.common.webservice.libraryserver.softlink.net/">
   <soapenv:Header/>
   <soapenv:Body>
      <ent:browse>
         <corpAlias>MyLibrary</corpAlias>
         <clientAliasEnc>555</clientAliasEnc>
         <clientPasswordEnc>555</clientPasswordEnc>
         <entityName>Client</entityName>
         <!--1 or more repetitions:-->
         <uuids>0567e7a6c0a8142e01fe48c2f8538e93</uuids>
         <loadConstant></loadConstant>
      </ent:browse>
   </soapenv:Body>
</soapenv:Envelope>
```

Example reply:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:browseResponse xmlns:ns2="http://entity.common.webservice.libraryserver.softlink.net/">
<return><![CDATA[
<net.softlink.libraryserver.webservice.common.entity.EntityAccessImpl_-BrowseResult>
  <valueObjects class="list">
    <Client>
      <uuid>0567e7a6c0a8142e01fe48c2f8538e93</uuid>
      <updateSearchIndex>true</updateSearchIndex>
      <corpId>0</corpId>
      <altPhone></altPhone>
      <email></email>
      <yearUuid>320d5b87c0a8142e0103a0327782c735</yearUuid>
      <userRoleUuid>7821d456c0a8142d013f3b3296a7aca6</userRoleUuid>
      <mailTitle></mailTitle>
      <gender>F</gender>
      <expiryDate class="sql-timestamp">2007-12-31 0:00:00.0</expiryDate>
      ...
</net.softlink.libraryserver.webservice.common.entity.EntityAccessImpl_-BrowseResult>]]></return>
      </ns2:browseResponse>
   </S:Body>
</S:Envelope>
```

Retrieving images:

Binary data is transferred in base64 representation (see: http://en.wikipedia.org/wiki/Base64). Here is a sample request to retrieve an image:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ent="http://entity.common.webservice.libraryserver.softlink.net/">
   <soapenv:Header/>
   <soapenv:Body>
      <ent:browse>
         <corpAlias>MyLibrary</corpAlias>
         <clientAliasEnc>nnn</clientAliasEnc>
         <clientPasswordEnc>nnn</clientPasswordEnc>
         <entityName>File</entityName>
         <uuids>dcf89567c0a814360024f9db7ff67539</uuids>
         <loadConstant></loadConstant>
      </ent:browse>
   </soapenv:Body>
</soapenv:Envelope>
```

And here is the reply with the bytes for the openbook.png file base64 encoded in the fileData XML element:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:browseResponse xmlns:ns2="http://entity.common.webservice.libraryserver.softlink.net/">
         <return><![CDATA[<net.softlink.libraryserver.webservice.common.entity.EntityAccessImpl_-
BrowseResult>
  <valueObjects class="list">
    <File>
      <uuid>dcf89567c0a814360024f9db7ff67539</uuid>
      <timeChanged class="sql-timestamp">2007-07-19 15:34:45.157</timeChanged>
      <timeEntered class="sql-timestamp">2007-07-19 15:34:45.157</timeEntered>
      <whoChanged>43527bbac0a814360172b4a693728c4b</whoChanged>
      <whoEntered>43527bbac0a814360172b4a693728c4b</whoEntered>
      <updateSearchIndex>true</updateSearchIndex>
      <corpId>0</corpId>
      <userCount>1</userCount>
      <size>287</size>
      <fileName>openbook.png</fileName>
<fileData>iVBORw0KGgoAAAANSUhEUgAAACAAAAAgCAMAAABEpIrGAAAAB3RJTUUH0gwFAiM2E5s8I
QAAAAlwSFlzAAALEgAACxIB0t1+/AAAAARnQU1BAACxjwv8YQUAAAAnUExURf///wAAAAAloCAgMDA
wMfHxtfV0s/OzP/78Pfz6u/s5Ofk3t/d2AcrTwUAAAABdFJOUwBA5thmAAAAbklEQVR42u2TSw6AIA
xEGSjyvf95rbQhjehaTXwhgZm87opznwHCXeZnLrX1rhXQe6slq8IXqbClo0HaVKChwBshwiEawS8C
gX7h5YIwBc1TQBCAIZisQpjwENkoggXhlOeCCtzwsVyt99N/bGEHlawMuPZdyX8AAAAASUVORK5CYII=
</fileData>
    </File>
  </valueObjects>
  <outer-class />
</net.softlink.libraryserver.webservice.common.entity.EntityAccessImpl_-BrowseResult>]]></return>
      </ns2:browseResponse>
   </S:Body>
</S:Envelope>
```

2. Create

Create a new database record of the given type.

***Arguments**:*

corpAlias: The alias of the corporation on this system we'll load from.

clientAliasEncZ: The alias of the Web Services user on this system whose login should be used. Should be RSA encrypted depending on parameter 1390.

clientPasswordEnc: The password of clientAlias. Should be RSA encrypted depending on parameter 1390.

entityName: Indicates the type of entity to perform the operation on.

valueObjectXML: Contains values to be applied to the new record. Must include the new uuid primary key.

Return a new value object in XML representation.

***Example request**:*

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ent="http://entity.common.webservice.libraryserver.softlink.net/">
   <soapenv:Header/>
   <soapenv:Body>
      <ent:create>
         <corpAlias>SaharaDB</corpAlias>
         <clientAliasEnc>nnn</clientAliasEnc>
         <clientPasswordEnc>nnn</clientPasswordEnc>
         <entityName>Catalog</entityName>
```

```
        <valueObjectXml><![CDATA[
  <net.softlink.libraryserver.data.CatalogValueObject>
    <title>title7</title>
  </net.softlink.libraryserver.data.CatalogValueObject>]]></valueObjectXml>
      </ent:create>
   </soapenv:Body>
</soapenv:Envelope>
```

***Example reply**:*

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:createResponse xmlns:ns2="http://entity.common.webservice.libraryserver.softlink.net/">
         <return><![CDATA[<Catalog>
  <uuid>666c9be6c0a8164e00ded9453ed3b06b</uuid>
  <timeChanged>07/12/2009</timeChanged>
  <timeEntered reference="/Catalog/timeChanged" />
  <whoChanged>f283add3c0a814300187ceb3a7fb98f6</whoChanged>
  <whoEntered>f283add3c0a814300187ceb3a7fb98f6</whoEntered>
  <updateSearchIndex>true</updateSearchIndex>
  <corpId>0</corpId>
  <secured>false</secured>
  <bibliographicTypeUuid>4530bb5cc0a814280161718984fe5cd3</bibliographicTypeUuid>
  <catalogId>544087</catalogId>
  <gmdUuid>e0edc847c0a81445018c1f36bd2ffb7d</gmdUuid>
  <sortableTitle>title8</sortableTitle>
  <title>title8</title>
  <isTemporaryLoanCatalog>false</isTemporaryLoanCatalog>
  </Catalog>]]></return>
      </ns2:createResponse>
   </S:Body>
</S:Envelope>
```

3. Delete

Delete a database record.

***Arguments**:*

corpAlias: The alias of the corporation on this system we'll load from.

clientAliasEnc: The alias of the Web Services user on this system whose login should be used. Should be RSA encrypted depending on parameter 1390.

clientPasswordEnc: The password of clientAlias. Should be RSA encrypted depending on parameter 1390.

entityName: Indicates the type of entity to perform the operation on.

Uuid: The record primary key.

Returns whether or not the operation was successful. To be successful the entity must exist and must not be referenced from another entity. A budget also cannot be deleted if there is an existing total or balance.

***Example request**:*

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ent="http://entity.system.webservice.libraryserver.softlink.net/">
   <soapenv:Header/>
   <soapenv:Body>
      <ent1:delete xmlns:ent1="http://entity.common.webservice.libraryserver.softlink.net/">
```

```
         <corpAlias>MyLibrary</corpAlias>
         <clientAliasEnc>nnn</clientAliasEnc>
         <clientPasswordEnc>nnn</clientPasswordEnc>
         <entityName>Catalog</entityName>
         <uuid>666c9be6c0a8164e00ded9453ed3b06b</uuid>
      </ent1:delete>
   </soapenv:Body>
</soapenv:Envelope>
```

Example reply:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:deleteResponse xmlns:ns2="http://entity.common.webservice.libraryserver.softlink.net/">
         <return>true</return>
      </ns2:deleteResponse>
   </S:Body>
</S:Envelope>
```

4. Invoke

Perform a generic business level operation.

***Arguments**:*

corpAlias: The alias of the corporation on this system we'll load from.

clientAliasEnc: The alias of the Web Services user on this system whose login should be used. Should be RSA encrypted depending on parameter 1390.

clientPasswordEnc: The password of clientAlias. Should be RSA encrypted depending on parameter 1390.

operationName: Identifies the name of the operation to invoke.

operationArgs: Container for operation attributes.

Returns the result of the operation, depending on operationName

To receive an acquisitions order pass "ReceiveOrder" as operationName, purchaseOrderUuid as the first element of operationArgs and numCopies as the second element of operationArgs, for example:

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ent="http://entity.common.webservice.libraryserver.softlink.net/">
   <soapenv:Header/>
   <soapenv:Body>
      <ent:invoke>
         <corpAlias>MyLibrary</corpAlias>
<clientAliasEnc>5478b265e1640008213bd93d68e59883c5b03afb0e727c648ad9c283cd6430a9725a52b2c7f68b58195
d54038f7d475e24c2f092c144f377261aac876c4d8aef347c11cad6d38331ee031f01a50ba6e75cc692f42b9d4c566110d2
01463e46872f5e2e7c91f3129374cb28d182174a09665c5636befbe1c2f8feefec9d233037</clientAliasEnc>
<clientPasswordEnc>78f2e380fbb7dc6134c756c1681fd26e49fa83989e9e4fbba7d1781f936bda42fd1850043bfa9794
64f62b10a1b43a0a20839c4058d81879bb18c8713fb42fcaaaa0e09b71cf4fe737f8ef13dda1e29264d23cd260838b258cb
9fe6fe8232a90347106f9ddee170180ce34fc3fee02c90cda25ab114adb6c21db6d6b1e8611c7</clientPasswordEnc>
         <operationName>ReceiveOrder</operationName>
         <!--1 or more repetitions:-->
         <operationArgs>3a034c65c0a81430010c52a265d2a3e3</operationArgs>
         <operationArgs>1</operationArgs>
      </ent:invoke>
```

```
  </soapenv:Body>
</soapenv:Envelope>
```

The above call will return the newly created uuid for the received order if successful, otherwise an error message.

5. Query

Find an entity using the given search criteria.

***Arguments**:*

corpAlias: The alias of the corporation on this system we'll load from.

clientAliasEnc : The alias of the Web Services user on this system whose login should be used. Should be RSA encrypted depending on parameter 1390.

clientPasswordEnc: The password of clientAlias. Should be RSA encrypted depending on parameter 1390.

entityName: Indicates the type of entity to perform the operation on.

Search: The user query to run.

maxResultCount: limits the number of search results to be returned

Returns a list of UUID's matching the given query.

***Example request**:*

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ent="http://entity.common.webservice.libraryserver.softlink.net/">
   <soapenv:Header/>
   <soapenv:Body>
      <ent:query>
         <corpAlias>My Library</corpAlias>
         <clientAliasEnc>nnn</clientAliasEnc>
         <clientPasswordEnc>nnn</clientPasswordEnc>
         <entityName>CostCentre</entityName>
         <search></search>
         <maxResultCount>100</maxResultCount>
      </ent:query>
   </soapenv:Body>
</soapenv:Envelope>
```

***Example reply**:*

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:queryResponse xmlns:ns2="http://entity.common.webservice.libraryserver.softlink.net/">
         <return>066e50fbc0a8140b004d2e1c254681bd</return>
         <return>086b6721c0a8142e0119eb859decb3f5</return>
         <return>086bafe2c0a8142e0119eb859d8fd092</return>
         <return>0ddcfc06c0a8142e01b6ea2e124b7b36</return>
         <return>22fb3badc0a8140b0029eae5614b0954</return>
         <return>313b9717c0a8142e01a9dbac343a2463</return>
         <return>3c5ae6ccc0a8140b00b74c940c9ae438</return>
         <return>4b30f088c0a814a10069d5bc8634efb3</return>
         <return>4b314f71c0a814a10069d5bc9ed13158</return>
         <return>c24d2032c0a8142e01777e846f49a3ba</return>
         <return>cc1edcd2c0a81430000a85aecb419025</return>
```

```
      </ns2:queryResponse>
   </S:Body>
</S:Envelope>
```

6. Update

Update the given entity using values from the given value object. Returns the updated entity in the form of an XML string.

***Arguments**:*

corpAlias: The alias of the corporation on this system we'll load from.

clientAliasEnc: The alias of the Web Services user on this system whose login should be used. Should be RSA encrypted depending on parameter 1390.

clientPasswordEnc: The password of clientAlias. Should be RSA encrypted depending on parameter 1390.

entityName: Indicates the type of entity to perform the operation on.

valueObjectXML: Contains values to be applied to the new record.

Returns the updated value object in XML representation.

***Example request**:*

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ent="http://entity.system.webservice.libraryserver.softlink.net/">
   <soapenv:Header/>
   <soapenv:Body>
      <ent1:update xmlns:ent1="http://entity.common.webservice.libraryserver.softlink.net/">
         <corpAlias>My Library</corpAlias>
         <clientAliasEnc>nnn</clientAliasEnc>
         <clientPasswordEnc>nnn</clientPasswordEnc>
         <entityName>Client</entityName>
         <valueObjectXml><![CDATA[<net.softlink.libraryserver.data.ClientValueObject>
  <uuid>0ffa8d21c0a8164e013c2a9521ff01a8</uuid>
  <middleInitial>D</middleInitial>
</net.softlink.libraryserver.data.ClientValueObject>]]></valueObjectXml>
      </ent1:update>
   </soapenv:Body>
</soapenv:Envelope>
```

***Example reply**:*

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:updateResponse xmlns:ns2="http://entity.common.webservice.libraryserver.softlink.net/">
         <return><![CDATA[<Client>
  <uuid>0ffa8d21c0a8164e013c2a9521ff01a8</uuid>
  <timeChanged>07/12/2009</timeChanged>
  <timeEntered class="sql-timestamp">2009-11-20 15:04:56.097</timeEntered>
  <whoChanged>f283add3c0a814300187ceb3a7fb98f6</whoChanged>
  <whoEntered>0fd6357bc0a8164e013c2a95b1e18b9a</whoEntered>
  <updateSearchIndex>true</updateSearchIndex>
  <corpId>0</corpId>
  <notes></notes>
  <address></address>
  <userName></userName>
  <fax></fax>
```

```
  <altPhone></altPhone>
  <email></email>
  <userRoleUuid>a7f1340cc0a814280061736e3bed1dc3</userRoleUuid>
  <mailTitle></mailTitle>
...
  <middleInitial>D</middleInitial>
</Client>]]></return>
      </ns2:updateResponse>
   </S:Body>
</S:Envelope>
```

7. ChangedEvent

Called to signal to a remote system that a change to an entity has occurred. The message receiver can then use the generic API to respond to the event as required.

***Arguments**:*

corporationId: Identifies the corporation related to the event.

entityName: the name of the entity that has changed

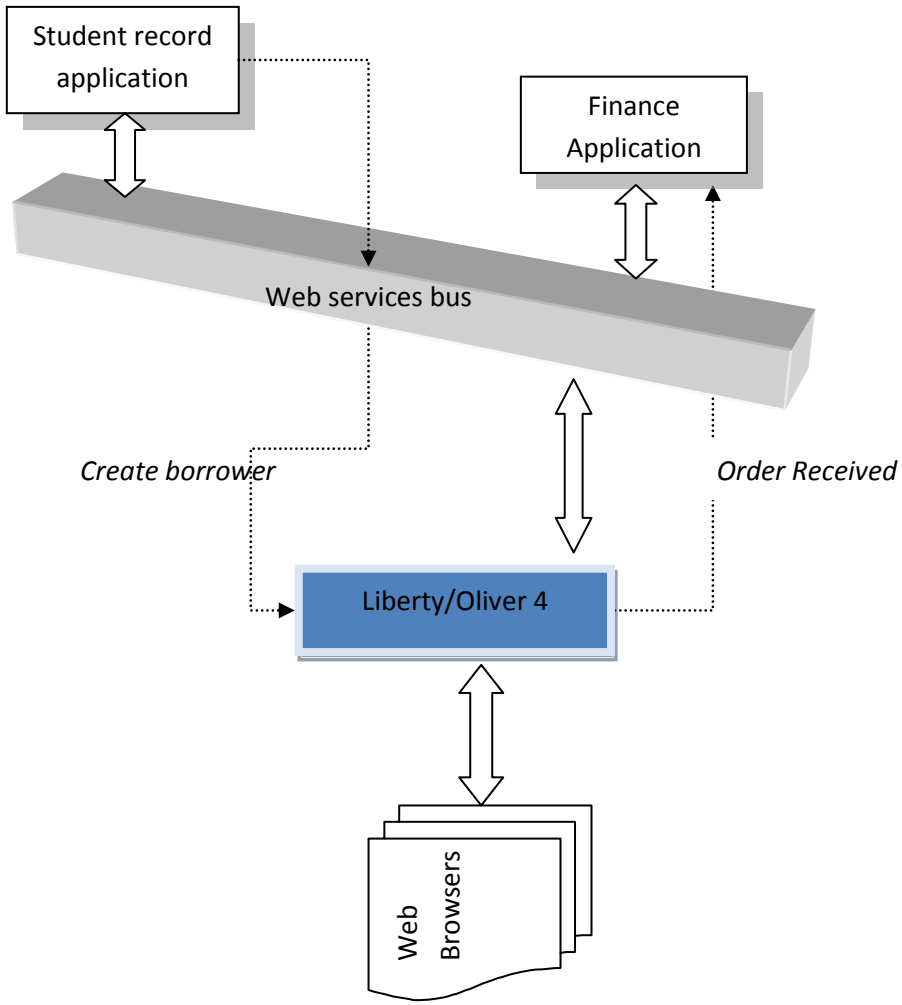
entityUuid: the primary key of the database row that has changed

```
void changedEvent(String corporationId, String entityName, entityUuid);
```

This message is called when an OrderReceived entity has been created or updated during the order receive business operation.

OPAC Search Web Services Expected Workflow

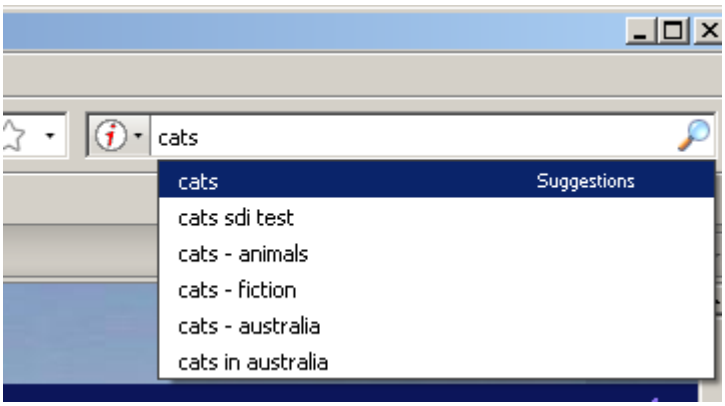
The following diagram illustrates a possible Web Services deployment scenario.



Open Search API Setup and Usage

The Open Search module implements version 1.1 of the Open Search specification. This module is available from v4.2 build 6.0 patch 9 onwards.

1. Searching from the browser search box



Allows a user to perform basic OPAC searching from the web browser search box. The search result is controlled by the OPAC system parameters. The Search plugin can be installed when the user visits the OPAC page.

2. Integration with MS SearchServer 2008

To integrate with MS Search Server 2008, a template import file can be dynamically generated using the following URL:

http://<host>:<port>/<context>/files/xml/openSearchFldTemplate.jsp?corporation=<corpalias>

Where <corpalias> matches the l3_alias field in the l3_corporation table and without any quote characters. If this is entered incorrectly you will simply get no SearchServer search results.

Save the exported result to a local file, and then import the file in SearchServer under the following path:

Search Server > Search Administration > Federated Locations > Manage Federated Locations > Import Location.

The following translation provides the customization of the search server import file.

searchserver.opensearch.title	Library: {0}
searchserver.opensearch.description	OpenSearch for library: {0}
searchserver.opensearch.channel.title	Results from Library: {0}
searchserver.opensearch.channel.title.top	Top Results from Library: {0}

Parameter 0 is a place holder for the corporation alias.

The following translation keys provide the customization of the rss result. Note the search server and browser may discard those values.

rss.opensearch.title	Search Result for {0}
rss.opensearch.description	Search Results for {0}
rss.opensearch.suggestion.description	{0} results

Parameter 0 is a place holder for the search term

The following system parameter must be setup correctly in order to form the correct URL in the SearchServer 2008 federated search results page: Host #1373, Port #1363, Protocol #1364. These parameters are also used by various other application components.

As can be seen in the following OpenSearch results screenshot, each result row consists of a Title, Description and a search engine URL.

When searching against Liberty, the description row is based on one of the following catalogue fields where the first non blank field encountered will be used: “Note”, “Abstract”, “Subject”.

Security

1. SSL

Web Services may be deployed using SSL over HTTPS to enhance security by ensuring network messages are not exchanged as plain text. The mechanisms to deploy Web Services with SSL are much the same as those for deploying the application user interface with SSL. Please refer to JBOSS documentation for further details.

2. Borrower Details Encryption

As an alternative to using SSL, you can protect borrower details passed into the API with the use of RSA encryption. This will simplify the deployment configuration over the use of SSL at the expense of a more complicated client implementation.

The following information is required in order to create the encrypted Web Services API arguments.

Public key modulus:

114480070560403827014835088066919947342593202818993977912622149979224374358595278172726419548621771092107446473960930993494950603755925261042475520201924359787800712238206015140388882558090302293311526325614078307688811737168400984140574272817850197922350289145284502582561245817415735208656528952682996460571

Public key exponent:

65537

Example borrower encryption using PHP*:*

The following demonstrates how borrower encryption could be implemented in the PHP language. Note that the application RSA decryption function expects a series of numeric characters represented as a hex string.

```
<?php
  include("rsa.class.php");
  $RSA = new RSA();
  $exponent = "65537";
  $modulus = "11448...."
  $client_alias =
    strhex($RSA->encrypt("administrator", $exponent, $modulus));
  $client_pw = strhex($RSA->encrypt("password", $exponent, $modulus));
  $client = new SoapClient("http://localhost:8080/oliver/OpacAccess?wsdl");
  $result = $client->query($corp_alias, $client_alias,
        $client_pw, $search, 25);
....
```

Capacity Planning

A number of performance factors must be considered for a successful Web Services deployment.

Note that the Web Services module has not yet been evaluated in a performance sensitive environment.

1. Server utilisation

Web Services incur similar system resource consumption to a client browser interaction for a similar type of request.

2. Network utilisation

The bandwidth requirements of the Web Services module depend on the frequency and scope of client requests.

Reponses from the OPAC specific API tends to be larger than the generic API due to build-in loadParameters used to return a broad selection of information. For example, the result from getBorrowerDetails() could be around 150KB.

You can limit the amount of returned information in the generic browse() API with careful use of the loadParameter argument to prevent unnecessary information from being returned.

A Web Service result has unnecessary white space removed but is otherwise not compressed.

3. Implementation model

Frequent polling may impact system performance in a similar way to running multiple frequent system interactions. The impact will be dependent on the interaction performed.

Currently the ChangeEvent API can be used to inform another application when an OrderReceived entity has been created or updated during the order receive business operation. With further development this could be extended to cover other internal system events.

Sample Java Encryption Program

```
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.security.GeneralSecurityException;
import java.security.Key;
import java.security.KeyFactory;
import java.security.interfaces.RSAPublicKey;
import java.security.spec.RSAPublicKeySpec;
import javax.crypto.Cipher;
import javax.crypto.CipherOutputStream;

public class SlkEncrypt {

  private static final BigInteger SLK_KEY_MODULUS = new
BigInteger("114480070560403827014835088066919947342593202818993977912622149979224374358595278172726
419548621771092107446473960930993494950603755925261042475520201924359787800712238206015140388882558
090302293311526325614078307688811737168400984140574272817850197922350289145284502582561245817415735
208656528952682996460571");  private static final BigInteger SLK_PUBLIC_KEY_EXPONENT = new
BigInteger("65537");

  /**
   * Program entry point
   */
  public static void main(String[] args) throws Exception {
    if (args.length != 1) {
      System.err.println("Usage: SlkEncrypt <term>");
    }
    else {
      System.err.println(new SlkEncrypt().encrypt(args[0]));
    }
  }

  /**
   * Return a hexadecimal formatted string of the given bytes.
   */
  private String byteArrayToHex(byte[] bytes) {
    StringBuilder sb = new StringBuilder(bytes.length * 2);
    for (int i = 0; (i < bytes.length); i++) {
      byte b = bytes[i];
      int j = b & 0xFF;
      String s = Integer.toHexString(j);
      if (s.length() < 2) {
        sb.append('0');
      }
      sb.append(s);
    }
    return sb.toString();
  }

  /**
   * Runs a cipher with the given name, using the given key and mode,
   * over the object and returns the result as a byte array.
   *
   * @param bytes To cipher.
   * @param key The key to use with the algorithm.
   * @param algorithmName The name of the cipher algorithm to use.
   * This should be one defined in the Java Cryptography Architecture.
   * @param MODE One of the modes from <code>javax.crypto.Cipher</code>
   * @return object, having been run through the cipher.
   */
  private byte[] cipher(byte[] bytes, Key key, String algorithmName,
                        final int MODE) {
    Cipher cipher = null;
    try {
      cipher = Cipher.getInstance(algorithmName);
```

```
    cipher.init(MODE, key);
  }
  catch (GeneralSecurityException e) {
    throw new IllegalArgumentException("No encryption algorithm exists
       with name [" + algorithmName + "]");
  }

  ByteArrayOutputStream outputStream = new ByteArrayOutputStream(bytes.length);
  CipherOutputStream cipherOutputStream =
     new CipherOutputStream(outputStream, cipher);
  try {
    cipherOutputStream.write(bytes);
  }
  catch (IOException e) {
    throw new RuntimeException(e);
  }
  finally {
    try {
      cipherOutputStream.close();
    }
    catch (IOException e) {
      throw new RuntimeException(e);
    }
  }

  byte[] cipheredObject = outputStream.toByteArray();
  return cipheredObject;
}


/**
 * Digitally signs a given object with the given privateKey, using the
 * given algorithm.
 *
 * @param bytes To sign.
 * @param privateKey The key to sign it with.
 * @param algorithmName The name of the signing algorithm to use.
 * This should be one defined in the Java Cryptography Architecture.
 * @return object as bytes, signed.
 */
private byte[] encrypt(byte[] bytes, Key key, String algorithmName) {
  return cipher(bytes, key, algorithmName, Cipher.ENCRYPT_MODE);
}

private String encrypt(String object) throws Exception {
  byte[] encrypted = encrypt(object.getBytes(),
      getSlkPublicKeyRSA(), "RSA");
  return new String(byteArrayToHex(encrypted));
}

/**
 * Creates an RSAPublicKey object, given the modulus and exponent
 * for the key
 *
 * @param modulus Of the key
 * @param exponent Of the key
 * @return The RSA key.
 */
private RSAPublicKey getPublicKeyRSA(BigInteger modulus,
        BigInteger exponent) throws Exception

  RSAPublicKeySpec publicKeySpec = new
    RSAPublicKeySpec(modulus, exponent);
  KeyFactory keyFactory = KeyFactory.getInstance("RSA");
  RSAPublicKey publicKey = (RSAPublicKey)
    keyFactory.generatePublic(publicKeySpec);
  return publicKey;
}
```

```
  /**
   * @return RSAPublicKey The public RSA key used for internal
   *  encryption of data.
   */
  private RSAPublicKey getSlkPublicKeyRSA() throws Exception {
    return getPublicKeyRSA(SLK_KEY_MODULUS, SLK_PUBLIC_KEY_EXPONENT);
  }
}
```