Microsoft

accenture

Danish Government Infostructurebase

API - SOAP User guide

Prepared by:
Simon T. Riemann

Index

Index 1
Introduction 2
Getting started 2
Example 1 Create SOAP request using XML Spy 3
Example 2 Upload to Repository SOAP API 5
Example 3 Find business using UDDI SOAP request 5
Repository APIs 6
    Error messages 6
    Documentmanager APIs 7
    Subscriptionservice APIs 11
    Objectstatemanager APIs 13
UDDI 14
    Find 15
    Getting details 16
    Save 17
    Delete 18
Metadata XSD 19
XSD schemes for XML generated by APIs 19
References 23

Introduction

Registered users may design and use SOAP API and WebDAV clients to use and extend the **Infostructurebase** functionality.

This user guide gives a description and overview of the SOAP APIs supplied within the **Infostructurebase** for the **Repository** and **UDDI**. The APIs for the **Repository** facilitates upload and download of documents and metadata (both SOAP and WebDAV) and management of subscriptions. The APIs for the **UDDI** are standard APIs facilitating searching and publishing of standards and services within the UDDI registry.

The definition and creation of a SOAP client is not part of this user guide. However as an example XML Spy version 5 enterprise edition is used as a third party tool using both the SOAP APIs and WebDAV. XML Spy may additionally be used as a schema development tool. For a detailed description of WebDAV access see the WebDAV user guide.

Each API is subject to a number of different input and output arguments. These arguments are described in the API reference.

As the **Infostructurebase** uses authentication for both the **UDDI** and **Repository** the use of SOAP APIs requires a registered user name and password. The UDDI registry does however supply inquiry APIs for unregistered users.

Users of this user guide are:

- **Unregistered users**
  Unregistered users may access the public **UDDI** inquire APIs.
- **Registered users**
  Registered users may access all **Repository** and **UDDI** APIs.

The structure of this user guide is as follows:

A short introduction on how to access and use the APIs followed by an API reference.

The API reference contains descriptions of the **Repository** APIs used for uploading and downloading documents from the **Repository** and the management of subscriptions and document status. Next the **UDDI** APIs used for finding, saving and deleting businesses, services, bindings, and tModels are presented.

The functions are presented by a short description along with the input syntax. For a complete description of the **UDDI** SOAP APIs consult the references listed in the back.

Getting started

Connect to the Web services for the **UDDI** and the **Repository** at:

http://isb.oio.dk/uddi/inquire

http://isb.oio.dk/uddi/publish

The UDDI APIs are divided in two sections. One for inquire APIs (e.g. findbusiness, getbusiness detail) and one for publishing APIs (e.g. savebusiness, deletebusiness) and one for publishing APIs. The latter requires the use of UDDI authentication in order to get permission to publish.
It is possible to use inquire APIs without being logged on using:

http://isb.oio.dk/uddipublic/inquire

This enables unregistered users to browse the UDDI registry.

The **Repository** APIs (all require a username and password) are available at:

http://isb.oio.dk/oioservice/service/documentmanager.asmx

http://isb.oio.dk/oioservice/service /subscriptionservice.asmx

http://isb.oio.dk/oioservice/service /objectstatemanager.asmx

Connect using your **Infostructurebase** username and password against basic authentication.

The **Repository** WSDL files may be retrieved in the usual way by extending the addresses above with ".asmx?wsdl". The UDDI WSDL files are accessible from

http://uddi.org/wsdl/inquire_v1.wsdl

http://uddi.org/wsdl/publish_v1.wsdl

All WSDL files are also found in the last part of this guide.

To assist you in developing clients for the UDDI you may find it useful to download the Microsoft UDDI SDK available as a stand alone for COM/COM+ development environments or for the VS.NET.
Inspire IT among others offers a Java client library implementing the UDDI V1.0 and V2.0 data structures and programmers API.

For the development of a client for the **Repository** you simply use this guide and the WSDL file.

Example 1
Create SOAP request using XML Spy

This example illustrates how to create a SearchDocument SOAP request to the **Repository** and analyzing the result using XML Spy version 5 Enterprise version.

From the SOAP menu select **Create SOAP request**. The WDSL file location dialog appears (see Figure 1).

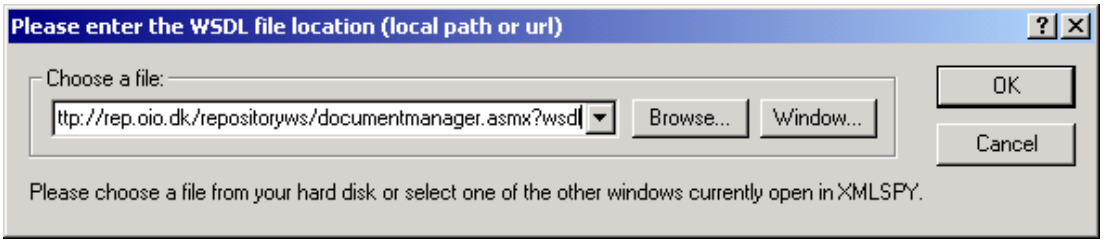

Figure 1 - WSDL file location dialog

Enter the location of the documentmanager web service wsdl file:
http://rep.oio.dk/repositoryws/documentmanager.asmx?wsdl and click **OK**.

When prompted enter your user name and password. The list of document manager APIs is shown for you to select the API to call (see Figure 2).

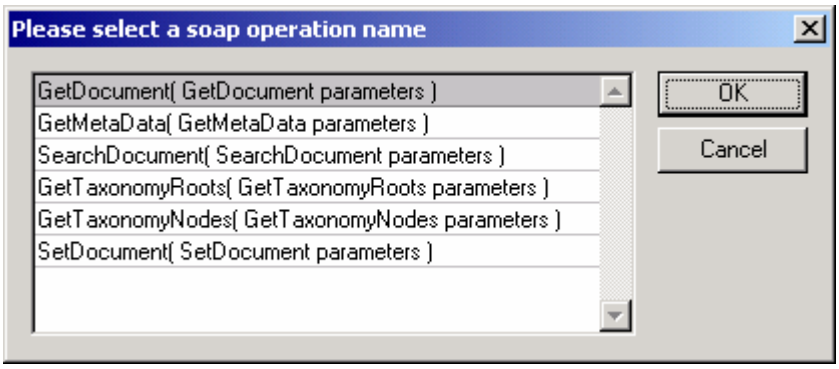

Figure 2 - API list

Select the SearchDocument SOAP operation and click **OK**. The SOAP envelope and body are now displayed in the main window (see Figure 3).

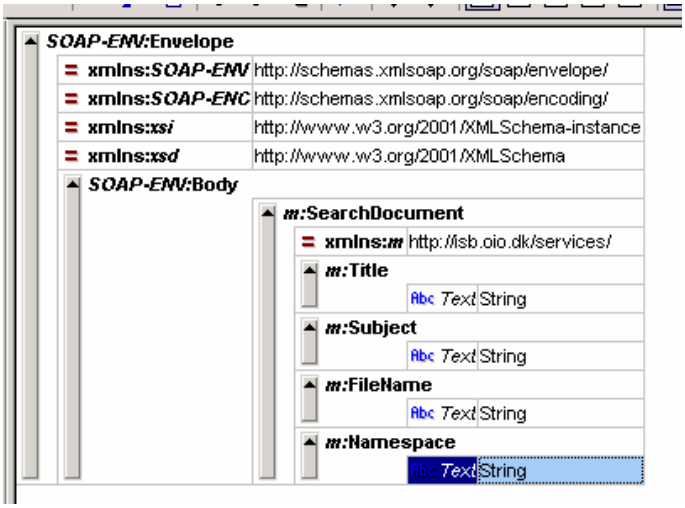

Figure 3 – SOAP envelope and body

To search for a document clear the pre-filled “String” text next to the Title, Subject, Filename and Namespace elements and enter the title or part of the title to search for. In the SOAP menu select **Send request to server**. The response is shown in a new window. Depending on the result of the search the response contains zero, one or more results. See Figure 4 for an example of a response containing one result.

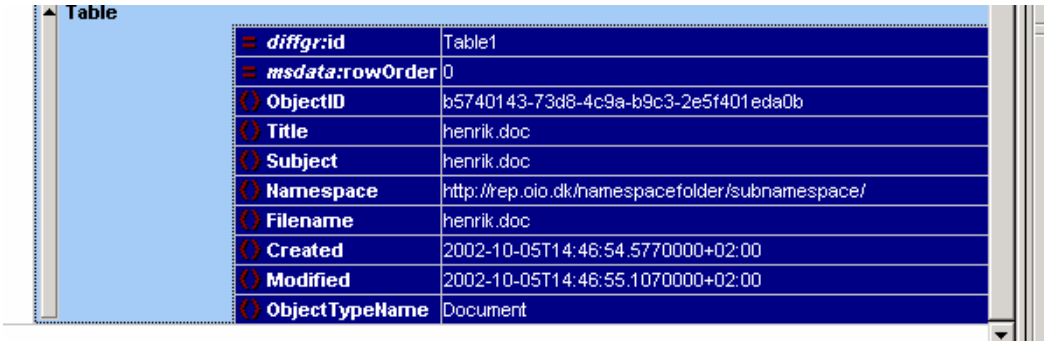

Figure 4 – Part of the SearchDocument SOAP response

Example 2
Upload to Repository SOAP API

This example illustrates how to upload a document and its metadata to the **Repository** using *SetDocument* and *SetMetadata* followed by the retrieval of metadata from the **Repository** using *GetMetaData*.

To invoke the APIs, you send SOAP messages with the appropriate body content.

The *SetDocument* API uploads the document "myfile.xml" to the namespace "mynamespace". The last parameter is the actual document (a byte array).

```
<SetDocument xmlns="http://rep.oio.dk/">
<documentNamespace>http://rep.oio.dk/xmlr.dk</documentNamespace>
      <DocumentNamespace>mynamespace</DocumentNamespace>
      <DocumentName>myfile.xml</DocumentName>
      <DocumentBytes>…</DocumentBytes>
</SetDocument>
```

The *SetMetaData* API sets the metadata for the document located at the URL:"http://rep.oio.dk/xmlr.dk/". The last metadata string is not complete and is only included as an illustration.

```
<SetMetadata xmlns="http://rep.oio.dk/">
<documentURL>http://rep.oio.dk/xmlr.dk/description.txt</documentURL>
<metadata> xml </metadata>
</SetMetadata>
```

The *GetMetaData* API gets the metadata just set for the document with the URL "http://rep.oio.dk/xmlr.dk/description.txt". The Metadata is returned as a string.

```
<GetMetadata xmlns="http://rep.oio.dk/">
      <documentURL>
    http://rep.oio.dk/xmlr.dk/description.txt
      </documentURL>
</GetMetadata>
```

Example 3
Find business using UDDI SOAP request

This example illustrates a search for service providers named "ShowBusiness1" with a discovery URL of "http://www.someOperator".

To invoke the APIs you send SOAP messages with the appropriate body content.

```
<find_business generic='1.0' xmlns="urn:uddi-org:api_v1">
<name>ShowBusiness1</name>
<discoveryURLs>
<discoveryURL useType="businessEntity">
http://www.someOperator
</discoveryURL>
</discoveryURLs>
</find_business>
```

The *find_business* API call returns a businessList on success. This structure contains information about each matching business, and summaries of the businessServices, including service projections, exposed by the individual businesses. If no arguments are passed, a zero-match result set will be returned.

```
<businessList generic="1.0" operator="uddi.sourceOperator" trun-
cated="true"
xmlns="urn:uddi-org:api_v1">
<businessInfos>
<businessInfo businessKey="F5E65…" >
<name>ShowBusiness1</name>
<serviceInfos>
<serviceInfo serviceKey="3D45…">
<name>Purchase Orders</name>
</serviceInfo>
</serviceInfos>
</businessInfos>
</businessList>
```

Repository APIs

The **Repository** supports SOAP APIs giving a registered user the ability to search for documents and taxonomies, uploading and downloading documents and metadata using a SOAP client. APIs for managing subscriptions and document status are also supplied.

The SOAP APIs available from the **Repository** are exposed at:

http://isb.oio.dk/oioservice/service/documentmanager.asmx

http://isb.oio.dk/oioservice/service /subscriptionservice.asmx

http://isb.oio.dk/oioservice/service /objectstatemanager.asmx

For some of the APIs you must supply your password as one of the parameters.

Error messages

Error messages are passed in standard SOAP fault messages with the following error codes and descriptions. All SOAP API are subject to throw standard SOAP exceptions when unexpected errors occurs.

Messages 500-614 are passed as strings.

- #100 Filename or namespace does not exist.
- #101 Invalid metadata.
- #102 Main document does not exist.
- #200 You do not have the rights to perform this action.
- #250 Failed to add subscription.
- #251 Failed to delete subscription.
- #252 Failed to get subscription.
- #253 No notifications found.
- #254 Failed to get notifications.
- #255 No modified documents found.

- #256 Failed to get modified documents.
- #257 No subscriptions found.
- #300 Invalid taxonomy node.
- #301 No Taxonomy nodes found.
- #400 Invalid state ID, object GUID or committee name.
- #401 Invalid Object GIUD.
- #500 File uploaded/updated.
- #501 Metadata updated.
- #502 Long metadata uploaded/updated.
- #503 Document added.
- #600 Object guid has an invalid format:
- #601 Problem with object guid:
- #602 No connection to database - please try again later.
- #603 Error. Please check that the GUID represents an existing object.
- #604 Document state: 'state' Committee: 'committee'
- #605 No state found. Please check that the GUID represents an existing object.
- #606 Error. No state found. Please check file and committee name.
- #607 The service is unable to complete your request - please try again later.
- #608 Object state updated successfully
- #609 Please supply values for all parameters.
- #610 Please check your date format:
- #611 Object state create failed. Please check that the GUID represents an existing object, and the specified new state represents a valid state.
- #612 Object state created successfully
- #613 Object state create failed. Please check that the filename and path represents an existing object, and that the specified new state represents a valid state.
- #614 Object state update failed. Please check that the GUID represents an existing object, and the specified new state represents a valid state.
- #800 Unexpected error.
- #900 No search result.
- #901 No long meta data found.
- #902 Document added.
- #903 Metadata updated.

Documentmanager APIs

FileAlreadyExists

Use this API to check if a document with a given name already exists in the namespace.

The return value is a boolean **true** if the file exists – **false** otherwise.

Syntax:
```
<FileAlreadyExists xmlns="http://isb.oio.dk/services/">
      <DocumentURL>
</FileAlreadyExists>
```

GetMetadata

Use this API to get the metadata for the document specified by the URL. The API returns the metadata as an XML string. The format of the string follows the metadata xsd structure (see Metadata XSD p.19).
Error messages: #100, #200, #800.

Syntax:
```
<GetMetaData xmlns="http://isb.oio.dk/services/">
      <DocumentURL>
</GetMetaData>
```

SetMetadata

Use this API to update the metadata xml file for the document specified by the documentURL using a metadata xml string (as defined in the metadata xsd). The return value is a message.

Error messages: #101, #102, #200, #800.
Messages: # 501.

Syntax:
```
<SetMetaData xmlns="http://isb.oio.dk/services/">
      <DocumentURL>
      <MetaData>
</SetMetaData>
```

GetDocument

Use this API to get the URL specified document as an array of bytes. The API returns the document in a byte array (base64binary encoding).
Error messages: #100, #200, #800.

Syntax:
```
<GetDocument xmlns="http://isb.oio.dk/services/">
      <DocumentURL>
</GetDocument>
```

SetDocument

Use this API to upload(update) a specified document to a namespace. The documentBytes input is the document represented in a byte array (base64binary encoding).

Error messages: #100, #200, #800.
Messages: #503.

Syntax:
```
<SetDocument xmlns="http://isb.oio.dk/services/">
      <documentNamespace/>
      <documentName/>
      <documentBytes/>
</SetDocument>
```

GetLongMetadataDocument

Use this API to get the URL specified long metadata document as an array of bytes. The

API returns the document in a byte array (base64binary encoding). The document name and document namespace refers to the document the long metadata is attached to.

Error messages: #100, #200, #800.

Syntax:

```
<GetDocument xmlns="http://isb.oio.dk/services/">
      <DocumentURL>
</GetDocument>
```

SetLongMetadataDocument

Use this API to upload a specified long metadata document to a namespace as an attachment to the document specified. The documentBytes input is the document represented in a byte array (base64binary encoding). The document name and document namespace refers to the document to attach to.

Error messages: #102, #200, #800.
Messages: #503.

Syntax:

```
<SetDocument xmlns="http://isb.oio.dk/services/">
      <DocumentURL/>
      <LongMetaDataDocumentName/>
      <LongMetaDataDocumentBytes/>
</SetDocument>
```

GetLongMetaDataDocumentList

Use this API to get URLs for long meta data documents attached to a document. The API returns an xml string containing all long meta data attached to the specific document.

Error messages: #800, #901.

The structure of the returned xml string is as follows:

<LongMetaDataDocuments> Root element
<LongMetaDataDocument> Each LongMetaDataDocument represents an attachment
<LongMetDataURL> The URL to the long meta data document
</LongMetaDataURL>
</LongMetaDataDocument>
</LongMetaDataDocuments>

Syntax:

<GetLongMetaDataDocumentList xmlns="http://isb.oio.dk/services/">
<DocumentURL>
</GetLongMetaDataDocumentList>

SearchDocument

Use this API to search for a document. You may search by document title, subject (based on metadata), filename and namespace. The API returns an xml string containing all documents matching the search criteria.

Error messages: #900.

The structure of the returned xml string is as follows:

<SearchResult> Root element.

<Document> Each table element represents a search match.
<Title> The document title.
<Subject> The document subject.
<DocumentURL> The URL to the document

<Created> The date of creation of the document.
<Modified> The date of the last modification of the document.
</Document>
</SearchResult>

Syntax:

```
<SearchDocument xmlns="http://isb.oio.dk/services/">
      <Title/>
      <Subject/>
      <Filename/>
      <Namespace/>
      <LastUpdate/>
      <Classification/>
      <TaxonomyKey/>
      <CustomMeta/>
      <ValueOfCostumMeta/>
      <ApprovedInCommittee/>
      <InHearingInCommittee/>
</SearchDocument>
```

```
Remarks:
<LastUpdate/> is in the form yyyy-mm-dd.
<TaxonomyKey/> is the key of a taxonomy node or taxonomy root.

Only one of the <ApprovedInCommittee> and <InHearingInCommittee> are used
in the search. If both are assigned values, the <ApprovedInCommittee>
value is used.
```

GetTaxonomyRoots

Use this API to get a list of the taxonomy roots. Use the key value of these roots to further explore the taxonomy using the GetTaxonomyNodes method.
The API takes no parameters, but returns a taxonomy node list with the following elements for each root node:

Error messages: #800.

<TaxonomyList> Root element
<Taxonomy> Each Taxonomy element represents a taxonomy root.
<Key> The node GUID.
<Name> The node name.
<Description> The node description.
</Taxonomy>
</TaxonomyList>

Syntax:

```
<GetTaxonomyRoots xmlns="http://isb.oio.dk/services/">
</GetTaxonomyRoots>
```

GetTaxonomyNodes

Use this API to get a list of taxonomy child nodes related to a parent node. The method takes the parent node GUID as input and returns only the first level nodes. To explore the taxonomy in depth call this method recursively.

The returned string is an xml string with the following structure:

<TaxonomyNodes> Root element
<Node> Each Node element represents a node
<Key> The node GUID.
<Taxonomy> The GUID of the parent node.
<Name> The node name.
<Description> Node description
</Node>
</TaxonomyNodes>

Or one of the following error messages: #300, #301.

Syntax:
```
<GetTaxonomyNodes xmlns="http://isb.oio.dk/services/">
      <Key/>
</GetTaxonomyNodes>
```

Subscriptionservice APIs

SetDocumentSubscription

Use this API to add a subscription to a document in a namespace. Select whether to subscribe to the document only or the document and all dependencies. A Subscriptiontype value of 0 indicates document only. A Subscriptiontype value of 1 indicates document and dependencies.
The return value is a boolean **true** for succes and **false** for failure.

Error messages: #250.

Syntax:
```
<SetDocumentSubscription xmlns="http://isb.oio.dk/services/">
      <DocumentURL/>
      <SubscriptionType/>
</SetDocumentSubscription>
```

SetNamespaceSubscription

Use this API to add a subscription to a namespace.

The return value is a boolean **true** for succes and **false** for failure.

Error messages: #250.

Syntax:
```
<SetNamespaceSubscription xmlns="http://isb.oio.dk/services/">
       <Namespace/>
</SetNamespaceSubscription>
```

DeleteDocumentSubscription

Use this API to delete a document subscription.

The return value is a boolean **true** for succes and **false** for failure.

Error messages: #251.

Syntax:
```
<DeleteDocumentSubscription xmlns="http://isb.oio.dk/services/">
       <DocumentURL/>
</DeleteDocumentSubscription>
```

DeleteNamespaceSubscription

Use this API to delete a namespace subscription.

The return value is a boolean **true** for succes and **false** for failure.

Error messages: #251.

Syntax:

```
<DeleteNamespaceSubscription xmlns="http://isb.oio.dk/services/">
        <Namespace/>
</DeleteNamespaceSubscription>
```

GetDocumentNotifications

Use this API to get an xml string with a list of all notifications on your subscribed documents since date of last login. The format of the returned string is as follows:

<DocumentNotifications> The root element
<DocumentNotification> The tag DocumentNotification represents a notification
<DocumentURL /> The URL to the document
<Description /> The notification
<NotificationDate /> The date of the notification
</DocumentNotification>
</DocumentNotifications>

Error messages: #253, #254.

Syntax:

```
<GetDocumentNotifications xmlns="http://isb.oio.dk/services/">
        <Date />
</GetDocumentNotifications>
```

GetNamespaceNotifications

Use this API to get an xml string with a list of all notifications on your subscribed namespaces since date of last login. The format of the returned string is as follows:

<NamespaceNotifications> The root element
<NamespaceNotification> The tag NamespaceNotification represents a notification
<DocumentURL /> The URL to the namespace
<Description /> The notification
<NotificationDate /> The date of the notification
</NamespaceNotification>
</NamespaceNotifications>

Error messages: #253, #254.

Syntax:

```
<GetNamespaceNotifications xmlns="http://isb.oio.dk/services/">
        <Date />
</GetNamespaceNotifications>
```

GetSubscriptions

Use this API to get an xml string with a list of all your subscriptions. The format of the

returned string is as follows:

<AllSubscriptions> The root element (no value)
<Documents> The root element for document subscriptions (no value)
<Subscriptions> The document subscriptions (no value)
<DocumentURL /> The URL to the subscribed document
</Subscriptions>
</Documents>
<Namespaces> The root element for namespace subscriptions (no value)
<Subscriptions> The namespace subscriptions (no value)
<Namespace /> The URL to the subscribed namespace
</Subscriptions>
</Namespaces>
</AllSubscriptions>

Error messages: #252, #257.

Syntax:

```
<GetSubscriptions xmlns="http://isb.oio.dk/services/">
</GetSubscription>
```

GetModifiedDocuments

Use this API to get an xml string with a list of subscribed items (documents) changed since a given date. The date is of type dateTime. The format of the returned string is as follows:

<DocumentsModified> The root element (no value)
<Document> Each Document tag represents a modified document (no value)
<DocumentURL /> The URL to the document
<Modified /> Date when the document was modified
<MetaDataModified /> Date when the meta data was modified
</Document>
</DocumentsModified>

Error messages: #255, #256.

Syntax:

```
<GetModifiedDocuments xmlns="http://isb.oio.dk/services/">
        <Date/>
</GetModifiedDocuments>
```

Objectstatemanager APIs

GetStateByDocumentURL

Use this API to get all the states of a document by the URL of the document. An XML string is returned. The ID of the state type can be translated as follows:

1: Approved
2: Rejected
3: In hearing
4: Obsolete

5: Archived
6: Pending (approval)

Result:

<DocumentStates> The root element
<DocumentState> Each DocumentState element represents a state in a committee
<Committee /> The committee
<Comment /> A comment
<Date /> Date when the state was created or updated
<StateTypeID /> The state of the document in the committee
</DocumentState>
</DocumentStates>

Error messages: #602, #606.

Syntax:

```
<GetStateByDocumentURL xmlns="http://isb.oio.dk/services/">
      <DocumentURL>
</GetStateByDocumentURL>
```

SetStateByDocumentURL

Use this API to create or update a state belonging to a document. You must supply the following parameters:

DocumentURL is the fully qualified URL (namespace and document name concatenated) of the document. **Committee** is the name of the committee associated with the state. **Comment** may be any comment. D**ate** is the date of the submission. **StateTypeID** is the state (refer to GetStateByDocumentURL).

Error messages: #602, #609, #610, #612, #613, #614.

Return messages: #612

Syntax:

```
<SetStateByDocumentURL xmlns="http://isb.oio.dk/services/">
      <DocumentURL>
      <Committee>
      <Comment>
      <Date>
      <StateTypeID>
</SetStateByDocumentURL>
```

UDDI

The **UDDI** supports a number of functions for finding, getting, publishing and deleting items in the **UDDI**.

The UDDI APIs are sorted in inquire and publish APIs. The list of available APIs from the UDDI are exposed at:

http://isb.oio.dk/uddi/inquire

http://isb.oio.dk/uddi/publish

The public inquire APIs are available at:

http://isb.oio.dk/uddipublic/inquire

Find

Software that allows people to explore and examine data – especially hierarchical data – requires browse capabilities. The browse pattern characteristically involves starting with some broad information, performing a search, finding general result sets and then selecting more specific information for drill-down. The UDDI API specifications accommodate the browse pattern by way of the *find_xx* API calls. These calls form the search capabilities provided by the API and are matched with summary return messages that return overview information about the registered information that is associated with the inquiry message type and the search criteria specified in the inquiry.

A typical browse sequence might involve finding whether a particular business you know about has any information registered. This sequence would start with a call to *find_business*, perhaps passing the first few characters of a business name that you already know. This returns a businessList result (see example 2). This result is overview information (keys, names and descriptions) derived from the registered businessEntity information, matching on the name fragment that you provided.

If you spot the business you are looking for within this list, you can drill down into the corresponding businessService information, looking for particular service types (e.g. purchasing, shipping, etc) using the *find_service* API call. Similarly, if you know the technical fingerprint (tModel signature) of a particular software interface and want to see if the business you've chosen provides a web service that supports that interface, you can use the *find_binding* inquiry message.

find_business

Use this (inquire) API to locate information about one or more businesses. The API returns a businessList message.

Syntax:

```
<find_business [maxRows="nn"] generic="2.0" xmlns="urn:uddi-org:api_v2" >
    [<findQualifiers/>]
    [<name/> [<name/>]…]
    [<discoveryURLs/>]
    [<identifierBag/>]
    [<categoryBag/>]
    [<tModelBag/>]
</find_business>
```

find_service

Use this (inquire) API to locate specific services within a registered businessEntity. The API should return a serviceList message.

Syntax:

```
<find_service [businessKey="uuid_key"] " [maxRows="nn"] generic="2.0"
          xmlns="urn:uddi-org:api_v2" >
    [<findQualifiers/>]
    [<name/> [<name/>]…]
    [<categoryBag/>]
    [<tModelBag/>]
```

```
</find_service>
```

find_binding

Use this (inquire) API to locate specific bindings within a registered businessService. The API should return a bindingDetail message.

Syntax:

```
<find_binding serviceKey="uuid_key" [maxRows="nn"] generic="2.0"
          xmlns="urn:uddi-org:api_v2" >
    [<findQualifiers/>]
    <tModelBag/>
</find_binding>
```

find_tModel

Use this (inquire) API to locate one or more tModel information structures. The API should return a tModelList structure.

Syntax:

```
<find_tModel [maxRows="nn"] generic="2.0" xmlns="urn:uddi-org:api_v2" >
    [<findQualifiers/>]
    [<name/>]
    [<identifierBag/>]
    [<categoryBag/>]
</find_tModel>
```

Getting details

Once you have a key for one of the four main data you can use that key to access the full registered details for a specific data instance. The current UDDI data types are businessEntity, businessService, bindingTemplate and tModel. You can access the full registered information for any of these structures by passing a relevant key type to one of the *get_xx* API calls.

Continuing the example from the previous section on browsing, one of the data items returned by all of the *find_xx* return sets is key information. In the case of the business you are interested in, the businessKey value returned within the contents of a businessList structure can be passed as an argument to *get_businessDetail*. The successful return to this message is a businessDetail message containing the full registered information for the entity whose key value was passed. This will be a full businessEntity structure.

get_businessDetail

Use this (inquire) API to get the full businessEntity information for one or more businesses or organizations. The API should return a businessDetail message.

```
Syntax:
<get_businessDetail generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <businessKey/> [<businessKey/> …]
</get_businessDetail>
```

get_serviceDetail

Use this (inquire) API to get full details for a given set of registered businessService data. The API should return a serviceDetail message.

Syntax:

```
<get_serviceDetail generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <serviceKey/> [<serviceKey/> …]
</get_serviceDetail>
```

get_bindingDetail

Use this (inquire) API to get full bindingTemplate information suitable for making one or more service requests. The API should return a bindingDetail message.

Syntax:

```
<get_bindingDetail generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <bindingKey/> [<bindingKey/> …]
</get_bindingDetail>
```

get_tModelDetail

Use this (inquire) API to get full details for a given set of registered tModel data. The API should return atModelDetail message.

```
Syntax:
<get_tModelDetail generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <tModelKey/>
    [<tModelKey/> …]
</get_tModelDetail>
```

Save

save_business

Use this (publish) API to register new businessEntity information or update existing businessEntity information. Use this to control the overall information about the entire business. Of the save_xx API's this one has the broadest effect.

Syntax:

```
<save_business generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    <businessEntity/> [<businessEntity/>…]
</save_business>
```

save_service

Use this (publish) API to register or update complete information about a businessService exposed by a specified businessEntity.

Syntax:

```
<save_service generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    <businessService/> [<businessService/>…]
</save_service>
```

save_binding

Use this (publish) API to register new bindingTemplate information or update existing bindingTemplate information. Use this to control information about technical capabilities exposed by a registered business.

Syntax:

```
<save_binding generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    <bindingTemplate/> [<bindingTemplate/>…]
</save_binding>
```

save_tModel

Use this (publish) API to register or update complete information about a tModel.

Syntax:
```
<save_tModel generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    <tModel/> [<tModel/>…]
</save_tModel>
```

Delete

delete_business

Use this (publish) API to delete registered businessEntity information from the registry.

Syntax:
```
<delete_business generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    <businessKey/>
    [<businessKey/> …]
</delete_business>
```

delete_service

Use this (publish) API to delete an existing businessService from the businessServices collection that is part of a specified businessEntity.

Syntax:
```
<delete_service generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    <serviceKey/>
    [<serviceKey/> …]
</delete_service>
```

delete_binding

Use this (publish) API to remove an existing bindingTemplate from the bindingTemplates collection that is part of a specified businessService structure.

Syntax:
```
<delete_binding generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    <bindingKey/> [<bindingKey/> …]
</delete_binding>
```

delete_tModel

Use this (publish) API to hide registered information about a tModel. Any tModel hidden in this way is still usable for reference purposes and accessible via the get_tModelDetail message, but is simply hidden from find_tModel result sets. There is no way to actually cause a tModel to be deleted.

Syntax:
```
<delete_tModel generic="2.0" xmlns="urn:uddi-org:api_v2" >
    <authInfo/>
    <tModelKey/> [<tModelKey/> …]
</delete_tModel>
```

Metadata XSD

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://rep.oio.dk/isb.dk/xml/schemas/" targetName-
space="http://rep.oio.dk/isb.dk/xml/schemas/">
    <xs:element name="MetaData">
          <xs:complexType>
                <xs:sequence>

                      <xs:element name="MetaDataElement" minOccurs="0"
maxOccurs="unbounded">
                            <xs:complexType>
                                  <xs:sequence>

                                                <xs:element
name="MetaDataAttribute" minOccurs="0" maxOccurs="unbounded">
                                                      <xs:complexType>

    <xs:attribute name="name" type="xs:string" />

    <xs:attribute name="value" type="xs:string" />
                                                      </xs:complexType>
                                                </xs:element>

                                  </xs:sequence>

                                  <xs:attribute name="name"
type="xs:string" />
                                  <xs:attribute name="value"
type="xs:string" />

                            </xs:complexType>
                      </xs:element>
                </xs:sequence>
          </xs:complexType>
    </xs:element>
</xs:schema>
```

XSD schemes for XML generated by APIs

SearchDocuments

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDe-
fault="qualified">

    <xs:element name="SearchResult">
          <xs:complexType>
                <xs:sequence>
                      <xs:element name="Document" maxOccurs="unbounded">
                            <xs:complexType>
                                  <xs:sequence>
                                                <xs:element name="Title"
type="xs:string"/>
                                                <xs:element name="Subject"
type="xs:string"/>
                                                <xs:element name="DocumentURL"
type="xs:anyURI"/>
                                                <xs:element name="Created"
type="xs:dateTime"/>
                                                <xs:element name="Modified"
type="xs:dateTime"/>
                                  </xs:sequence>
                            </xs:complexType>
                      </xs:element>
```

```
                </xs:sequence>
            </xs:complexType>
    </xs:element>

</xs:schema>
```

GetTaxnonomyRoots

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDe-
fault="qualified">
    <xs:element name="TaxonomyList">
          <xs:complexType>
                <xs:sequence>
                      <xs:element name="Taxonomy" maxOccurs="unbounded">
                            <xs:complexType>
                                  <xs:sequence>
                                        <xs:element name="Key"
type="xs:string" />
                                        <xs:element name="Name"
type="xs:string" />
                                        <xs:element name="Description"
type="xs:string" />
                                  </xs:sequence>
                            </xs:complexType>
                      </xs:element>
                </xs:sequence>
          </xs:complexType>
    </xs:element>
</xs:schema>
```

GetTaxonomyNodes

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDe-
fault="qualified">
    <xs:element name="TaxonomyNodes">
          <xs:complexType>
                <xs:sequence>
                      <xs:element name="Node" maxOccurs="unbounded">
                            <xs:complexType>
                                  <xs:sequence>
                                        <xs:element name="Key"
type="xs:string"/>
                                        <xs:element name="Taxonomy"
type="xs:string"/>
                                        <xs:element name="Name"
type="xs:string"/>
                                        <xs:element name="Description"
type="xs:string"/>
                                  </xs:sequence>
                            </xs:complexType>
                      </xs:element>
                </xs:sequence>
          </xs:complexType>
    </xs:element>
</xs:schema>
```

GetLongMetaDataDocumentList

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDe-
fault="qualified">
    <xs:element name="LongMetaDataDocuments">
          <xs:complexType>
                <xs:sequence>
                      <xs:element name="LongMetaDataDocument" maxOc-
curs="unbounded">
                            <xs:complexType>
                                  <xs:sequence>
                                        <xs:element
name="LongMetaDataURL" type="xs:anyURI"/>
                                  </xs:sequence>
                            </xs:complexType>
                      </xs:element>
```

```
                </xs:sequence>
            </xs:complexType>
    </xs:element>
</xs:schema>
```

GetSubscription

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDe-
fault="qualified">
    <xs:element name="AllSubscriptions">
          <xs:complexType>
                <xs:sequence>
                      <xs:element name="Documents">
                            <xs:complexType>
                                  <xs:sequence>
                                        <xs:element
name="Subscriptions">
                                                      <xs:complexType>

    <xs:sequence>

    <xs:element name="DocumentURL" maxOccurs="unbounded"
type="xs:string"/>

    </xs:sequence>
                                                      </xs:complexType>
                                        </xs:element>
                                  </xs:sequence>
                            </xs:complexType>
                      </xs:element>

                      <xs:element name="Namespaces">
                            <xs:complexType>
                                  <xs:sequence>
                                        <xs:element
name="Subscriptions">
                                                      <xs:complexType>

    <xs:sequence>


    <xs:element name="Namespace" maxOccurs="unbounded" type="xs:string"/>

    </xs:sequence>
                                                      </xs:complexType>
                                        </xs:element>
                                  </xs:sequence>
                            </xs:complexType>
                      </xs:element>
                </xs:sequence>
          </xs:complexType>
    </xs:element>
</xs:schema>
```

GetNamespaceNotifications

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDe-
fault="qualified">
    <xs:element name="NamespaceNotifications">
          <xs:complexType>
                <xs:sequence>
                      <xs:element name="NamespaceNotification" maxOc-
curs="unbounded">
                            <xs:complexType>
                                  <xs:sequence>
                                        <xs:element name="Namespace"
type="xs:anyURI"/>
                                        <xs:element name="Description"
type="xs:string"/>
                                        <xs:element
name="NotificationDate" type="xs:dateTime"/>
                                  </xs:sequence>
```

```
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
</xs:schema>
```

GetModifiedDocuments

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDe-
fault="qualified">
    <xs:element name="DocumentsModified">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Document" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="DocumentURL"
type="xs:string"/>
                            <xs:element name="Modified"
type="xs:dateTime"/>
                            <xs:element
name="MetaDataModified" type="xs:dateTime"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

GetDocumentNotifications

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDe-
fault="qualified">
    <xs:element name="DocumentNotifications">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="DocumentNotification" maxOc-
curs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="DocumentURL"
type="xs:anyURI"/>
                            <xs:element name="Description"
type="xs:string"/>
                            <xs:element
name="NotificationDate" type="xs:dateTime"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

GetStateByDocumentURL

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDe-
fault="qualified">
    <xs:element name="DocumentStates">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="DocumentState" maxOc-
curs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Committee"
type="xs:string"/>
                            <xs:element name="Comment"
type="xs:string"/>
                            <xs:element name="Date"
type="xs:dateTime"/>
                            <xs:element name="StateTypeID"
```

```
type="xs:byte"/>
                                        </xs:sequence>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
</xs:schema>
```

References

http://www.w3.org

For useful and in-depth information about SOAP and other web related standards.

http://www.uddi.org/specification.html

Contains the complete UDDI API and data structure specification for UDDI version 2.

http://msdn.microsoft.com

Contains useful information about using UDDI.