# Microscope AIM Data Service

## Purpose

The MicroAIM Data Service is a caGrid 1.3 data service that manages (stores and provides) documents in the MicroAIM xml format. MicroAIM is an adaptation of the Annotation and Image Markup (AIM) information model in order to support specific requirements in Pathology and broader Microscopy image mark up and annotations.

## Reference Installation

A reference installation of the service is located here at the following URL.  
<http://tma01.cci.emory.edu:8080/wsrf/services/cagrid/MicroAIMDataService>

WSDL for the service is available at  
<http://tma01.cci.emory.edu:8080/wsrf/services/cagrid/MicroAIMDataService?wsdl>  
<http://tma01.cci.emory.edu:8080/wsrf/share/schema/MicroAIMDataService/MicroAIMDataService_bindings.wsdl>

## MicroAIM Requirements

1. A MicroAIM document provides a context in which the annotations and related information has meaning. We see two major types of queries: retrieve whole MicroAIM document for analysis and display; and retrieve attributes such as nuclei size for exploration of data in the data service.

2. MicroAIM documents are design to annotate one or more markups and associated observations linked via provenance. An image may have multiple associated AIM documents as additional analysis, human annotation, or derivations are performed. The documents themselves may be large. Efficiency in data retrieval from the backend database to the grid is therefore important.

3. MicroAIM data service should have standard caGrid service interface for it to easily interoperate in the caBIG environment. This interface needs to be able to support querying and submission of MicroAIM data over the grid.

## Architecture Design Rationale

[caGrid Introduce](http://cagrid.org/display/introduce/Home) is used to generate standard source code skeletons for all caGrid based services. Introduce's data service extension provides a single standard operation in the interface: query.

Pre-defined adaptor layers have been defined for both relational databases as well as xml databases. Relational databases are supported through [caCORE SDK](https://cabig.nci.nih.gov/tools/caCORE_SDK) and the Introduce caCORE SDK Data Service plugin. XML databases are supported through the [xService](https://web.cci.emory.edu/confluence/display/xmlds/caGrid+xService) caGrid Introduce Data Service plugin.

**caCORE SDK based grid services** has the property that it retrieve only the Target object specified in the [CQL](http://cagrid.org/display/dataservices/CQL) . The associated objects are not retrieved. This means that to reconstitute an AIM document, which has multiple associated elements for each ImageAnnotation or AnnotationOfAnnotation, multiple queries will need to be performed. The query results will then need to be matched to the correct annotation and the annotation reconstructed. [CQL2](http://cagrid.org/display/dataservices/CQL+2) will address the associated object materialization problem but that has not been released. At the moment, using caCORE SDK based grid service is an unscalable solution for document-centric, context-sensitive data. caCORE SDK based grid service has only the "query" method and no support for data submission.

**xService based grid service** translates CQL queries into XPath query statements and execute them against the backend xml database. The XPath query returns the subtree identified by the XPath. This means that all objects associated with the Target object are returned during a caGrid based query. xService also provides default implementation logic for "submit" in addition to "query", as well as the logic for utilizing caGrid Transfer for high performance data transfer for query and submission. Finally, xService implementation optimizes the data transfer from the database to the grid interface and the client via 1. query results are kept as xml and serialization and deserializaiton are avoided wherever possible, and 2. iterative retrieval of the query results from the database where possible.

xService make use of **caGrid Transfer** as an alternative method to SOAP for transporting xml documents. caGrid transfer is primarily a binary transport mechanism. The use of caGrid Transfer is to support the transmission of large xml documents that can cause excessive serialization and deserialization overheads. xService transports the xml documents in a compressed or uncompressed zip stream. The rationale for using a zip stream is to provide a way to aggregate multiple xml documents into the same caGrid Transfer context, thus avoiding the overhead of creating one transfer context per xml document to be moved.

Additional information about xService can be found [here](https://web.cci.emory.edu/confluence/display/xmlds/caGrid+xService)

## Service Architecture

MicroAIM Data Service was developed using the caGrid Introduce xService extension based on the requirement of maintaining document integrity and data context as maintained by the MicroAIM document.

A service created using the caGrid XService Extension leverages all standard caGrid service capabilities and contains the following components:

1. caGrid data service instance with CQL to XPath Query Translation: CQL queries will be translated into XPath queries. The results are returned to the client as XML documents, which may be optionally deserialized into Java beans.  
2. Native XML Database: The Backend native XML database provides the storage, indexing and querying of stored XML documents. For the reference implementation of the MicroAIM documents, IBM DB2 PureXML is used.

## APIs and Behaviors

The MicroAIM data service provides 4 standard operations: query, submit, queryWithTransfer, submitWithTransfer. All operations are caGrid compliant, SOAP based operations with autoboxing that binds java objects to underlying SOAP message payload in XML formats. This is done for both input parameters and output results for each operation. The APIs are described in terms of the bound java bean objects. Service side API utilizes standard operations and implementation logic from XService. The documentation below will show only WSDL segment in the service operations, and Java client API.

### query()

Purpose: the query function allows a client user to query the database for MicroAIM documents by example, as specified using CQL. The parameter (CQL) and the output (query results) are transmitted using SOAP messages.

#### Input Parameter

CQLQuery Object, constructed using class names, association names, and attribute names, as defined in the MicroAIM data model. A query contains a target class to retrieve and associated classes and attribute filter criteria.

#### Output Parameter

CQLQueryResults Object, containing instances of result objects as defined by the caGrid [specification for CQLQueryResults](http://www.cagrid.org/display/dataservices/CQL+Schemas) . Specifically, the results may be objects conforming to the MicroAIM data model, attribute tuples, or count of results.

#### Functional Description

The query operation takes a CQLQuery and performs the query against the XML database for matching XML elements (as mapped to Data Model classes). Results returned include XML elements matching the corresponding target class in the CQL.

#### API

Service WSDL: see wsdl link above.  
input parameter conforms to CQL schema, which is available here: <http://cagrid.org/display/dataservices/CQL+Schemas>  
output conforms to the CQLQueryResults schema, which is available here: <http://cagrid.org/display/dataservices/CQL+Schemas>

Java client API:  
**class**

edu.emory.cci.microaim.microaimdataservice.client.MicroAIMDataServiceClient

**operation**

publicgov.nih.nci.cagrid.cqlresultset.CQLQueryResults query(gov.nih.nci.cagrid.cqlquery.CQLQuery cqlQuery)throwsRemoteException, gov.nih.nci.cagrid.data.faults.QueryProcessingExceptionType, gov.nih.nci.cagrid.data.faults.MalformedQueryExceptionType

### submit()

Purpose: the submit function allows a client user to submit one or more MicroAIM documents to the database. The documents are submitted via SOAP messaging.

#### Input Parameter

The submit input parameter is an array of Strings. the content of the Strings are XML instance documents conformant to the MicroAIM data model in XSD representation.

#### Output Parameter

none.

#### Functional Description

The submit operation takes in a set of xml documents in MicroAIM format and insert them into the service. The service does not currently validate that the documents conform to the data model, nor does it currently enforce any policy on prevention of submitting duplicate documents to the data service. Note that only validate documents can be queried since the CQL is mapped to XPath that conform to the XSD.

#### API

Service WSDL: see wsdl link above.  
input parameter is a string[]

Java client API:  
**class**

edu.emory.cci.microaim.microaimdataservice.client.MicroAIMDataServiceClient

**operation**

publicvoid submit(java.lang.String[] xmls)throwsRemoteException, gov.nih.nci.cagrid.data.faults.QueryProcessingExceptionType, gov.nih.nci.cagrid.data.faults.MalformedQueryExceptionType

### queryWithTransfer()

Purpose: the queryWithTransfer function allows a client user to query the database for MicroAIM documents by example, as specified using CQL. The parameter (CQL) is transmitted using SOAP messages. The results are returned to the client via caGrid Transfer.

#### Input Parameter

CQLQuery: Constructed using class names, association names, and attribute names, as defined in the MicroAIM data model. A query contains a target class to retrieve and associated classes and attribute filter criteria.

#### Output Parameter

TransferServiceContextReference: a reference to a dynamically constructed grid service resource that holds the result data for retrieval. The reference object contains the URL for the Transfer Service Resource Context, which then can point to the results in a file.

#### Functional Description

The queryWithTransfer operation takes a CQLQuery object as input and performs the query against the XML database for matching XML elements (as mapped to Data Model classes). Result XML documents are packaged into a ZIP file and staged to a caGrid Transfer Service resource instance. The user interacts with the resource instance to retrieve the zip file. IT is the responsibility of the client to unzip the file to get the XML documents. Below is the sequence of actions for using queryWithTransfer to retrieve MicroAIM documents.

1. User invokes queryWithTransfer with CQL  
2. MicroAIM Service executes the CQL query and obtains results from the database in XML form  
3. MicroAIM Service packages the XML documents into a ZIP file  
4. MicroAIM Service stages the ZIP using a TransferServiceHelper for client retrieval  
5. TransferServiceHelper generates a TransferServiceContextReference for the staged ZIP file  
6. MicroAIM Service returns the TransferServiceContextReference to the client  
7. Client creates a TransferServiceClient using the resource context URL in the TransferServiceContextReference  
8. Client invokes TransferServiceClient operation to retrieve TransferDescription  
9. Client instantiates a TransferClientHelper with the TransferDescription, which contains a URL pointing to the actual file staged  
10. Client uses TransferClientHelper to retrieve the file from the file URL.  
11. Client decompresses the ZIP file to access the MicroAIM XML Documents

The sequence of events described above includes both the API of the MicroAIM service as well as the caGrid Transfer Service. The caGrid Transfer Service APIs are not described here.

#### API

Service WSDL: see wsdl link above.  
input parameter conforms to CQL schema, which is available here: <http://cagrid.org/display/dataservices/CQL+Schemas>  
output is a TransferServiceContextReference. Schema is here

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

<schema

targetNamespace=<http://transfer.cagrid.org/TransferService/Context/types>

xmlns:wsa=<http://schemas.xmlsoap.org/ws/2004/03/addressing>

xmlns:tns=<http://transfer.cagrid.org/TransferService/Context/types>

xmlns:xs=<http://www.w3.org/2001/XMLSchema>

xmlns=<http://www.w3.org/2001/XMLSchema>

xmlns:wsrbf=<http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-BaseFaults-1.2-draft-01.xsd>

elementFormDefault="qualified"

attributeFormDefault="unqualified">

<element name="TransferServiceContextReference">

<complexType>

<sequence >

<element ref="wsa:EndpointReference"></element>

</sequence>

</complexType>

</element>

</schema>

Java client API:  
**class**

edu.emory.cci.microaim.microaimdataservice.client.MicroAIMDataServiceClient

**operation**

publicorg.cagrid.transfer.context.stubs.types.TransferServiceContextReference queryByTransfer(gov.nih.nci.cagrid.cqlquery.CQLQuery cqlQuery)throwsRemoteException, gov.nih.nci.cagrid.data.faults.QueryProcessingExceptionType, gov.nih.nci.cagrid.data.faults.MalformedQueryExceptionType

### submitWithTransfer()

Purpose: the submit function allows a client user to submit one or more MicroAIM documents to the database. The documents are submitted via caGrid Transfer.

#### Input Parameter

None.

#### Output Parameter

TransferServiceContextReference: a reference to a dynamically constructed grid service resource that provides a URL for data upload. The reference object contains the URL for the Transfer Service Resource Context, which is a placeholder for a uploaded file.

#### Functional Description

The submitWithTransfer operation does not take an input. On invocation, it creates a caGrid Transfer Resource Context that is ready to receive the XML documents in a ZIP file. On receiving the ZIP file, a registered call-back function then unpack the ZIP file and stores the contained XML documents into the database. It is the responsibility of the client to zip the XML documents into a ZIP file. Below is the sequence of actions for using submitWithTransfer to upload MicroAIM documents.

1. User invokes submitWithTransfer  
2. TransferServiceHelper generates a TransferServiceContextReference for staging a ZIP file  
3. MicroAIM Service returns the TransferServiceContextReference to the client  
4. Client creates a ZIP file with all the MicroAIM documents to be submitted  
5. Client creates a TransferServiceClient using the resource context URL in the TransferServiceContextReference  
6. Client invokes TransferServiceClient operation to retrieve TransferDescription  
7. Client instantiates a TransferClientHelper with the TransferDescription, which contains a URL pointing to the place where the ZIP upload file is to be sent.  
8. Client invokes TransferClientHelper's operation to upload the ZIP file  
9. Client invokes "staged" operation on the TransferServiceClient to signal that the transfer is complete  
10. MicroAIM service's registered callback function is invoked to unpack the ZIP files and submit each contained MicroAIM documents to the database.

The sequence of events described above includes both the API of the MicroAIM service as well as the caGrid Transfer Service. The caGrid Transfer Service APIs are not described here.

#### API

Service WSDL: see wsdl link above.  
no input  
output is a TransferServiceContextReference. Schema is here

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

<schema

targetNamespace=<http://transfer.cagrid.org/TransferService/Context/types>

xmlns:wsa=<http://schemas.xmlsoap.org/ws/2004/03/addressing>

xmlns:tns=<http://transfer.cagrid.org/TransferService/Context/types>

xmlns:xs=<http://www.w3.org/2001/XMLSchema>

xmlns=<http://www.w3.org/2001/XMLSchema>

xmlns:wsrbf=<http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-BaseFaults-1.2-draft-01.xsd>

elementFormDefault="qualified"

attributeFormDefault="unqualified">

<element name="TransferServiceContextReference">

<complexType>

<sequence >

<element ref="wsa:EndpointReference"></element>

</sequence>

</complexType>

</element>

</schema>

Java client API:  
**class**

edu.emory.cci.microaim.microaimdataservice.client.MicroAIMDataServiceClient

**operation**

publicorg.cagrid.transfer.context.stubs.types.TransferServiceContextReference submitByTransfer()throwsRemoteException, gov.nih.nci.cagrid.data.faults.QueryProcessingExceptionType, gov.nih.nci.cagrid.data.faults.MalformedQueryExceptionType

## Important Notices

As the MicroAIM Data Service is implemented using a different backend technology than relational database driven caCORE SDK system, there are some important behavioral differences to note.

### CQL support

MicroAIM Data Service supports all the predicates and syntax of CQL. It currently does not support traversing bidirectional associations properly. While traversing towards leave nodes in the XML and XSD is fully supported, traversing towards the root node will not return results. This is a known problem that will be resolved.

MicroAIM Data Service does not implicitly map the "id" attribute to any unique identifier. It manages the MicroAIM documents as submitted and does not modify the documents to insert unique identifiers.

MicroAIM Data Service, when queried, returns the full subtree, i.e., associated objects, from the target object.

MicroAIM Data Service does not enforce removal of duplicate data elements at any level, whether root document element or otherwise.

### Security

The MicroAIM Data Service does not currently require user authentication. MicroAIM Data Service does not currently support authorization at service, operation, or data instance level.

## Installation Instructions

The installation instruction of the MicroAIM Data Service follows those for XService. This includes the following steps:

1. Install cagrid, tomcat, and caGrid Transfer service <http://wiki.cagrid.org/display/downloads/caGrid+1.3+Installation+Quick+Start>  
2. Install IBM DB2 PureXML database <https://web.cci.emory.edu/confluence/display/xmlds/IBM+DB2+Express-C+Installation+Guide>  
3. Configure the database <https://web.cci.emory.edu/confluence/display/xmlds/Configure+DB2+access>  
4. copy the db2config.xml into the MicroAIM Data Service etc directory.  
5. deploy MicroAIM Data Service into the Tomcat Container with

ant deployTomcat

6. start up Tomcat with

cd $CATALINA\_HOME bin/startup.sh

7. verify install using a web browser by going to <http://localhost:8080/wsrf/services/cagrid/MicroAIMDataService>