

User Manual
ISAAC.lib – 1.5.0
Document Revision A

Point of Contact
Christian Kerth

12th June 2014

Fraunhofer IOSB
Fraunhoferstraße 1
76131 Karlsruhe

Change Log

| Modified by | Date | Revision | Description of Modification |
|--------------------------------------|------------|----------|----------------------------------------------|
| Christian Kerth, Barbara Essendorfer | 12.06.2014 | A | Initial English Version for SW Version 1.5.0 |

Index

| | | |
|-------------------|-----------------------------------------------|-----------|
| 1 | Definitions and Abbreviations | 4 |
| 2 | Introduction | 5 |
| 3 | Connection via Java Library | 7 |
| 3.1 | Conceptual Design | 7 |
| 3.2 | CSD Client | 7 |
| 3.3 | WritingClient..... | 8 |
| 3.4 | ReadingClient..... | 11 |
| 3.5 | Result Handler | 14 |
| 3.6 | Integrated HTTP Server | 15 |
| 4 | Structure of the XML Files..... | 16 |
| 5 | Property Objects | 19 |
| 5.1 | LibraryConnectionConfig | 19 |
| 5.2 | ReadingClientProperties..... | 19 |
| 5.3 | WritingClientProperties..... | 19 |
| 6 | Known Compatibility Limitations | 20 |
| 6.1 | Legacy Systems..... | 20 |
| 6.2 | Related Files and Updates | 21 |
| 7 | Sources | 22 |
| Appendix A | Notes Regarding Usage..... | 23 |
| A.1 | System Time | 23 |
| A.2 | Network..... | 23 |
| A.3 | Parameters for Methods | 23 |
| A.4 | Integration in OSGi | 23 |
| Appendix B | – Example of a Metadata File..... | 24 |
| B.1 | ReadingClient..... | 24 |
| B.2 | WritingClient..... | 25 |
| Appendix C | – Example of Configuration Files | 27 |
| C.1 | WritingClient..... | 27 |
| C.2 | ReadingClient..... | 28 |
| Appendix D | – ISAAC.lib API Javadoc..... | 29 |

1 Definitions and Abbreviations

| | |
|-----------------|------------------------------------------------------------------------------------|
| CORBA | Common Object Request Broker Architecture |
| CSD | Coalition Shared Data |
| CSD-Server | Any CSD-Server based on STANAG 4559 |
| CSD Server | Research and development CSD-Server of Fraunhofer IOSB (including basic functions) |
| CSD PLUS Server | Operational usable CSD-Server of Fraunhofer IOSB (including additional functions) |
| GB | Gigabyte |
| IOR | Interoperable Object Reference |
| ISAAC | ISr Artifact Access Client |
| ISR | Intelligence, Surveillance, Reconnaissance |
| JDK | Java Development Kit |
| JRE | Java Runtime Environment |
| MAJIC | Multi-int All-source Joint ISR Interoperability Coalition |
| MB | Megabyte |
| NSILI | NATO Standard ISR Library Interface; see STANAG 4559 |
| STANAG | Standardization Agreement |
| STANAG 4559 | Identification of the STANAG the CSD/CSD PLUS Server is based on. |

2 Introduction

The CSD-Server stores so-called ISR products¹ (and is therefore also referred to as ISR product library). An ISR product consists of a file (e.g. an image, a video, a message or a GMTI scan) and associated metadata (e.g. the time or location at which the file was created).

The product files can have various different formats:

- Images: STANAG 4545 (NSIF)
- GMTI data: STANAG 4607
- Videos: STANAG 4609
- Messages: XML (predefined schema)
- ...

For specific product types, no information is added in the form of a product file. All information is contained in the metadata in that case. In addition to the metadata and the product file auxiliary files ("Related Files") may be attached, for example a lower-resolution overview image or a section of the video file.

These various products can then be associated with each other within the CSD-Server to allow a message to be associated directly with the underlying image, video or information requirement.

The product metadata complies with the data model of STANAG 4559 [1] or enhancements in the MAJLIC project. The data model itself is split into various groups (entities). An assignment of all attributes contained in the model is not mandatory: Many attributes are optional and can be assigned to increase the information content of the metadata.

For communication between client and server a CORBA²-based protocol is used (see STANAG 4559 [1]). This protocol is used to transfer both requests for reading or creating products as well as all metadata. Product files and Related Files are transferred between client and server using HTTP. This applies both to creating new and changing existing products, and to calling up products via the Order Manager. The required HTTP server is incorporated in the ISAAC.lib and starts up automatically as needed.

¹ ISR = Intelligence, Surveillance, Reconnaissance; an ISR product is thus a product of the field of intelligence, surveillance and reconnaissance.

² CORBA = Common Object Request Broker Architecture; a specification for object-oriented middleware.

The metadata is always passed to or returned by the ISAAC.lib in the form of XML data. The associated schemas can be based on rules automatically generated from the data model of the CSD-Server. Pregenerated schemas can be provided on request.

The following section provides an overview of the client's functions.

3 Connection via Java Library

3.1 Conceptual Design

The library is designed such as to provide separate interfaces for Read and Write access. These interfaces are instantiated by the respective factory to ensure that their implementation is isolated as effectively as possible.

Functionalities offered by both the read and the write interface are relocated to the CSDClient base interface. This interface has the sole function of retrieving the data model.

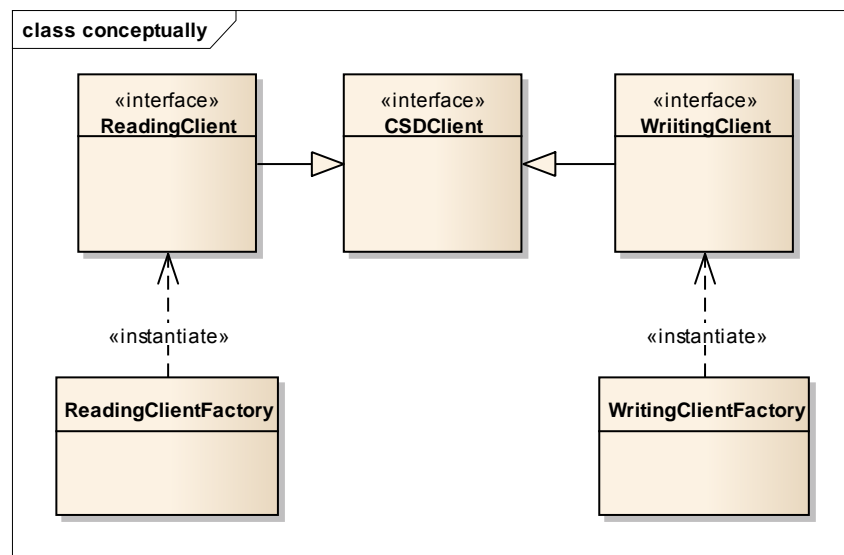


Figure 1: Conceptual Design of the Library Interfaces

The server connection required to perform the operations is established automatically in the background. Because the client does not keep a connection to the server open in the background, the connection is open only for the duration of the interaction.

3.2 CSD Client

The CSDClient interface provides common operations of the ReadingClient and the WritingClient.

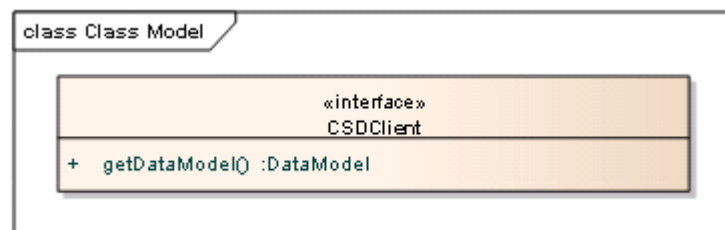


Figure 2: Operations of the CSDClient Interface

The only available operation is the retrieval of the server's data model, which can be used both internally for validating and transforming the transferred metadata, and for supporting the user in creating queries.

3.3 WritingClient

The WritingClient provides Write access to the CSD-Server. For generating new products or associations the client addresses the interfaces of the so-called Creation Manager in the server; for changing or deleting products or associations it addresses the interfaces of the Update Manager. For individual operations the Product Manager and the DataModel Manager are also used.

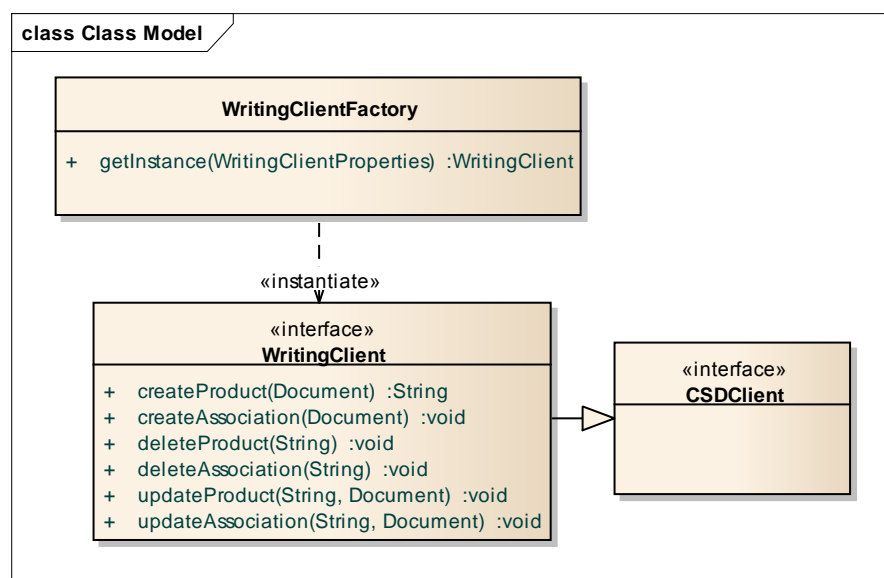


Figure 3: Operations of the WritingClient

To allow the WritingClient to complete its tasks, various initialization parameters are required. These parameters are summarized in the WritingClientProperties class. To access an instance of the WritingClient, you must first call up its factory, which then carries out a full initialization. When you receive the WritingClient, it is therefore already fully initialized and ready for use.

```

WritingClientProperties p = ...
WritingClient client = WritingClientFactory.createInstance(p);
  
```

Code 1: Creating a WritingClient Instance

This code snippet illustrates how to obtain the instance of the WritingClient. You can either fill in the WritingClientProperties yourself or generate the configuration object from a configuration file. For an example of this configuration file, see the Appendix of this document.

Automatic creation of the configuration object is carried out using the PropertyLoaders.

```
Properties p = ...  
WritingClientProperties wcp = PropertyLoader.loadWritingClientProperties(p);
```

Code 2: Generating a WritingClientProperties Instance

Once you have created an instance of the Writing Client, all of the commands described in the interface are available.

3.3.1 Creating an Artifact

To write an artifact to the CSD-Server, only the metadata and – for products – the product file are required. The product file is referenced in the metadata through corresponding attributes. Any Related Files can also be specified in the metadata.

The metadata passed to the WritingClient must have an XML data structure.

3.3.1.1 Products

```
public String createProduct(Document xmlMetadata) throws CreationFailedException
```

Code 3: Adding a new Product

On successful call, this method returns the value of attribute NSIL_CARD.identifier. This attribute provides you with globally unique access to this product.

If an error has occurred when adding the product, a CreationFailedException is thrown, the content of which may provide more detailed information about the cause of the error. Possible error causes include:

- No connection to the CSD-Server
- Attributes used do not exist in the CSD-Server
- Attribute cannot be set by the client
- Value of an attribute lies outside the allowed range

Files specified in the metadata must always point to the local file system. Files that are already on an HTTP or FTP server cannot be referenced. The files are transferred using the HTTP server incorporated in ISAAC.lib (see section 3.6 "Integrated HTTP").

For an example of product metadata see section 4 "Structure of the XML".

3.3.1.2 Associations

```
public void createAssociation(Document xmlMetadata) throws CreationFailedException
```

Code 4: Adding a new Association

Unlike the creation of a product, this method does not return a unique identification value, since this is not implemented in the CSD-Server's interface. The unique identifier can be determined only through a search within the transferred metadata.

If an error has occurred when adding the association, a `CreationFailedException` is thrown, the content of which provides more detailed information about the cause of the error. Possible error causes include:

- No connection to the CSD-Server
- Attributes used do not exist in the CSD-Server
- Attribute cannot be set by the client
- Value of an attribute lies outside the allowed range
- The products specified for the association do not exist

For an example of association metadata see section 4 "Structure of the XML".

3.3.2 Changing an Artifact

Artifacts already existing in the CSD-Server can be changed with the `ISAAC.lib`. The modification possibilities are limited, however: It is not possible, for example, to exchange the product file or to change attributes assigned by the server.

All other information that can be given for the `ISAAC.lib` when adding a product can be changed. These changes are then subject to the same validation checks that are applied when the artifact was created.

The following can be changed or added:

- Attributes assigned client-side
- Related Files
- Associations

3.3.2.1 Products

```
public void updateProduct(String id, Document newMetadata, Map<String, String> relFiles)
                                                                    throws UpdateFailedException
```

Code 5: Modifying existing Product Metadata

The input parameters of the method are the product's globally unique identifier that you have, for example, received as return value when adding the product, the new metadata, and a list of Related Files to be saved.

The new metadata must correspond to the required final state. Omitting values therefore corresponds to deleting attributes, and adding values corresponds to adding attributes. For changed values, the attribute values are adjusted accordingly. Unchanged values do not alter the attribute value. When transferring metadata, make sure that the metadata set by the server is not changed or omitted, since the library would interpret this as a modification or deletion. Because the server does not allow clients to modify this metadata, this would generally result in the request being rejected.

The list of Related Files is a list of changes. The key of the map must match the value of the `RELATED_FILE.fileType` to be modified. The value referenced by the key must be the absolute path to the file. The Related Files are added to the product if no Related File with this identifier is associated with the product yet. The Related File is replaced if the identifier already exists. Related Files can be deleted by setting a *null* value for the path (not a string, but a blank). Because the servers may partly use different dialects for handling Related Files, not every server may allow their modification.

Absolute paths must be specified as map values. The files are transferred using the HTTP server incorporated in ISAAC.lib (see section 3.6 “Integrated HTTP”).

3.3.2.2 Associations

```
public void updateAssociation(String id, Document newMetadata) throws UpdateFailedException
```

Code 6: Changing Existing Association Metadata

The input parameters are the globally unique identifier of the association and the new metadata to be saved.

The metadata must be in its required final state. As when changing a product, ISAAC.lib compares the actual state with the desired target state and modifies the metadata accordingly.

3.3.3 Deleting an Artifact

Previously created artifacts cannot be fully deleted through the server interface. They can, however, be flagged obsolete to indicate to other users of the server content that this information should no longer be used.

3.3.3.1 Products

```
public void deleteProduct(String id) throws DeletionFailedException
```

Code 7: Deleting Products

The only parameter required for deleting a product is the product’s globally unique identifier.

3.3.3.2 Associations

```
public void deleteAssociation(String associationId) throws DeletionFailedException
```

Code 8: Deleting Associations

The only required parameter for deleting an association is the association’s globally unique identifier.

3.4 ReadingClient

The ReadingClient implements Read access to the CSD-Server. There are two types of Read procedure:

- In the first Read procedure, the request is sent to the server, which returns information about the current content. This is referred to as a query.
- The second Read procedure returns, in addition to the current content related to the query, new and changed relevant artifacts at intervals. This is referred to as a subscription.

For found artifacts, the metadata is returned in the form of XML data for convenient processing. The format of the metadata largely corresponds to the XML format for setting or changing products, but can also contain attributes assigned by the CSD-Server.

Product files and the Related Files are accessed via the addresses in the metadata.

To obtain an instance of the ReadingClient, the corresponding factory must, as for the WritingClient, be called up. The only parameter for the factory is ReadingClientProperties. This configuration structure contains all the information necessary for operating the ReadingClient. The instance of the returned ReadingClient is fully initialized and you can use it straight away.

```
ReadingClientProperties p = ...
ReadingClient client = ReadingClientFactory.createInstance(p);
```

Code 9: Creating a ReadingClient Instance

As for the WritingClient, the ReadingClient allows the configuration object to be created from a configuration file. As for the WritingClient, you must use the PropertyLoader utility class for this purpose.

Having created an instance of the ReadingClient, you can use the operations described below. For a detailed description of the commands, see the following sections.

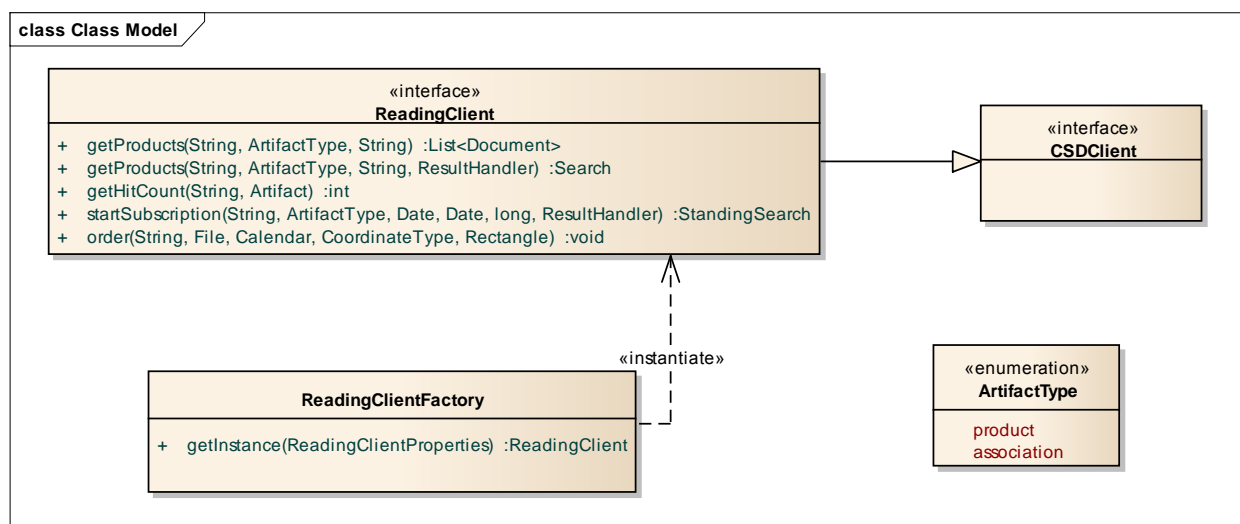


Figure 4: Interfaces of the ReadingClient

As indicated by their names, the commands offer the two search methods mentioned above. In addition you can determine the number of hits returned by a search and have image sections returned by the server.

3.4.1 Query

The simple query provides users with two variants that are each intended for a specific type of application. The first variant returns all results that match the query in the form of a single list. The list is therefore returned only once all results have been gathered. The second variant is designed to return subsets even when further results can still be retrieved from the CSD-Server.

In both cases the search filter (in the form of a BQS string), the type of artifact (product or association), and the sort order (attribute name) are transferred. In the second variant a ResultHandler, to which incoming results are sent must be passed in addition. If an error occurs during initiation of the query, a SearchFailedException is thrown and the query is canceled on the server. Possible causes of this error can be:

- Error in the query
- Non-existent view
- Non-existent attribute

When using the variant with the ResultHandler, a search object, with which the search can be terminated prematurely is also returned.

3.4.2 Subscription

As mentioned at the beginning of this section, the big difference between a query and a subscription is that subscriptions return regular updates about changes in the database in addition to a one-time snapshot of the CSD-Server. These can include, for example, the addition of new artifacts or changed metadata.

To limit the lifespan of a subscription, it has a defined start and end time. This also allows subscriptions to be defined for processing at some point in the future. Because the specified times are processed by the server, you must make sure that the system times of client and server are synchronized.

The server periodically scans its database for further products that match the query between the start and end times. The time interval at which the queries take place can be specified with parameter `millisecondBetweenSearches`. It is recommended not to deviate too much from the usual five seconds, as servers have different allowable value ranges and may be overloaded by excessively small time values.

Subscriptions are created in much the same way as normal queries. The differences are that subscriptions have no sorting attribute but, instead, feature the additional subscription start/end time and search frequency (parameter `millisecondsBetweenSearches`).

For the start time we recommend that you choose a time in the recent past to make sure that the subscription will be processed immediately when you issue the request.

The return value for this method is a `StandingSearch` object. With this object you can cancel the subscription prematurely, pause execution, and access state information. It does not provide information about the found data.

3.4.3 Ordering

In the CSD-Server language the term “order” refers to access to a part of a product file. Ordering is used when a client system requires only a small section of a high-resolution image and not the full data.

An order consists of the unique product identifier, the name of the target file to which the result is to be written, and the coordinates of the image section to be generated. With the additionally available parameter `needByDate` the server can be informed about the time by which the image section is required. However, very few servers support this function (this also applies to the IOSB CSD PLUS Server).

The coordinates of the image section can be specified in various ways: Firstly, the servers support calculation using the pixel coordinate system of the saved image. The total size of the stored image is available to the user through the ICHIPB extension of the overview or through the metadata.

Alternatively, the image section can be specified using world coordinates (longitude coordinates in geodetic reference system WGS84). However, since not all servers use the same information to translate world coordinates into pixel coordinates (e.g. IGEOLO/SENSORB or observance of altitude data), the use of the pixel coordinate system is recommended.

CoordinateType “wholeArea” is used to receive the entire product. This can be compared with passive FTP downloads, in which the connection is established by the server and is therefore rarely used. If you need the entire file, it is advisable to use direct access via the address in the metadata.

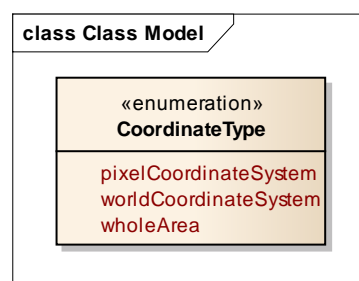


Figure 5: Possible Types of Coordinates

The files are transferred using the HTTP server incorporated in ISAAC.lib (see section 3.6 “Integrated HTTP”). The results of the order process are automatically written to the desired location in the file system.

3.5 Result Handler

The ResultHandler listens for events or results that can occur in the CSD-Server following a query.

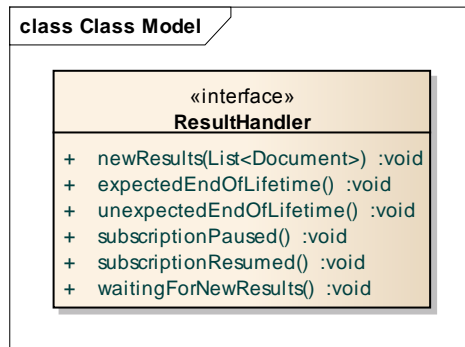


Figure 6: ResultHandler Interface

If you implement your own ResultHandler, you must implement all methods. These methods must not throw any exceptions, as this would negatively impact the queries from the ReadingClient. Events subscriptionPaused and subscriptionResumed are triggered only for StandingQueries; they are not used for simple queries.

3.6 Integrated HTTP Server

For file transfers, ISAAC.lib automatically starts the integrated HTTP server on the configured port, and closes it again when the interaction is completed. If multiple files are uploaded at the same time (both from a single call and from multiple simultaneous calls), the HTTP server instance is used for all transmissions and closes only when the last file transmission has been completed.

4 Structure of the XML Files

All XML files transferred from and to ISAAC.lib are structured according to the following rule set.

- All XML elements and attributes as specified without namespace.
- STANAG 4559 entities are represented as XML elements.
 - The name of an element is the name of the entity without the prefix "NSIL_".
 - Sub-entities and STANAG 4559 attributes are represented as child elements in XML.
 - Entities without child entities or attributes must be omitted.
- STANAG 4559 attributes are represented as XML element.
 - The name of the element is the name of the attribute.
 - The element's value takes the form described in the table below.

| Data type STANAG | Representation |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BOOLEAN_DATA | Text with "true" or "false" |
| UCOS_RECTANGLE | Two XML sub-elements according to the following example, numbers as Double :: toString (): <lowerRightPoint> <latitude>16.707881</latitude> <longitude>37.111466</longitude> </lowerRightPoint> <upperLeftPoint> |

| | |
|----------------|----------------------------------------------------------------------------------------|
| | <latitude>16.71077</latitude> <longitude>37.108159</longitude> </upperLeftPoint> |
| UCOS_ABS_TIME | Text of format: YYYY-MM-DD'T'hh:mm:ss.sss'Z' Example: "2012-09-04T12:37:29.628Z" |
| FLOATING_POINT | Text in format Double::toString() |
| INTEGER | Text in format Long.toString() |
| TEXT | As Text, escaping required as for XML. |

Example for product:

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

```
<PRODUCT>
```

```
<CARD>
```

```
<dateTimeModified>2014-02-12T16:45:04.963Z</dateTimeModified>
```

```
<identifier>061e199c-9405-11e3-afde-001f1612f1e7</identifier>
```

```
<numberOfParts>1</numberOfParts>
```

```
<sourceDateTimeModified>2014-02-12T16:45:04.963Z</sourceDateTimeModified>
```

```
<sourceLibrary>IOSB CSD-Server 1</sourceLibrary>
```

```
<status>NEW</status>
```

```
</CARD>
```

```
[...]
```

```
</PRODUCT>
```

Example for association:

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

```
<ASSOCIATION>

  <SOURCE>

    <CARD>

      <identifier>##NSIL_CARD.identifier value of source product##</identifier>

    </CARD>

  </SOURCE>

  <DESTINATION>

    <CARD>

      <identifier>##NSIL_CARD.identifier value of destination product##</identifier>

    </CARD>

  </DESTINATION>

  <RELATION>

    <description>association</description>

    <relationship>FOLLOWS</relationship>

  </RELATION>

</ASSOCIATION>
```

5 Property Objects

The classes described in this section are used for generating the client ReadingClient/WritingClient objects of the respective factories.

5.1 LibraryConnectionConfig

This class describes the connection information to the CSD-Server. This information includes:

- The HTTP address at which the CORBA entry address is retrieved
- The access for the individual managers of the STANAG 4559 CORBA interface

For anonymous access, the access data must be specified with an empty string (in class UsernamePassword).

5.2 ReadingClientProperties

This class contains a LibraryConnectionConfig object and the port on which the integrated HTTP server is started on demand. The third and final component is the systemName, which is specified in searches as "user_info".

5.3 WritingClientProperties

In addition to the ReadingClientProperties, this class contains a LibraryConnectionConfig object, the port on which the integrated HTTP server is started on demand, and the systemName, which is specified as "user_info" for searches that are required to implement the operations of the WritingClient.

In addition, the client-side forced classifications are specified with allowedClassificationOnServer. In all interactions with the CSD-Server in which new products are added or existing ones modified, the presence of the classifications specified in the metadata (NSIL_SECURITY.classification and NSIL_METADATASECURITY.classification) in the configured set of values is checked.

Values validationProductPath and validationAssociationPath are defined in the WritingClientProperties class are no longer used and are defined only for downward compatibility. They can therefore have any value (including "zero" and empty string).

6 Known Compatibility Limitations

6.1 Legacy Systems

Some older servers do not exhibit the exact behavior expected by the library. To compensate for this, the library has so-called legacy options, which can be enabled globally for all connections.

```
LegacySupport.setEnabled(LegacyOption.IOR_WITH_WHITESPACES);
```

Code 10: Activation of a Legacy Option

If the client is connected to a server that supports data model v2.1, option `DATAMODEL_DOMAIN_STRINGLENGTH_0_AS_255` must be enabled. Otherwise valid metadata would be rejected by the clients during validation (in older model versions, length 0 was defined as unlimited). In newer data models this option has no effect.

The second option is intended for servers that append so-called whitespace characters (spaces, tabs, etc.) in the returned IOR¹. If this option is active even though the server does not append additional whitespace characters, it has no effect.

The third option is intended for cases in which the use of the JacORB on the client side causes problems because of its own class loader. In this case, the ORB that SUN supplies in the VM is used. This does not, however, have the same stability as the JacORB. The activation of this operation has no known effects on compatibility with servers, as it acts only on the client side.

The last option is intended for servers that do not support keyword "ALL" in the Product/Catalog/StandingQuery Manager. When this option is active, the client always transmits the complete list of all attributes contained in the corresponding view. Since this results in an overhead of several kilobytes per query, it is recommended to use this option only if the server does not support the keyword.

¹ IOR = Interoperable Object Reference; an entry address required for CORBA communication.

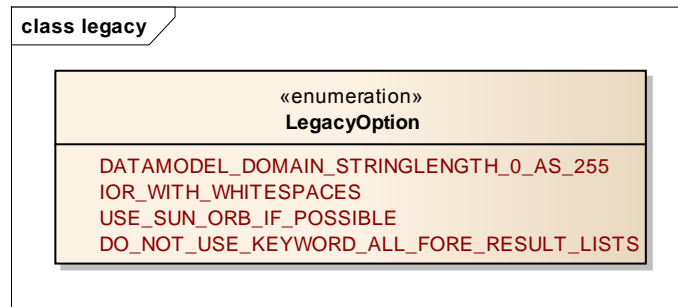


Figure 7: List of Available Legacy Options

6.2 Related Files and Updates

For both Related Files and UpdateDAGs the corresponding parts in the STANAG are not uniquely defined. This has the consequence that some servers classify the requests generated by the library as invalid. In rare cases the server may provide results other than those expected.

For using the library with the IOSB CSD PLUS Server no limitations are known.

7 Sources

- [1] NATO Standardization Agency, „Standardization Agreement (STANAG) 4559 Edition 3,“ 2010.
- [2] „GitHub,“ [Online]. Available: <https://github.com/openGDA/diamond-jacorb/tree/master/uk.ac.diamond.org.jacorb>. [Zugriff am 20.05.2014].

Appendix A Notes Regarding Usage

A.1 System Time

The library is written for use on a system whose time zone is set to UTC.

A.2 Network

The library automatically determines the system's network configuration. On systems with multiple actively used network interfaces problems with the integrated HTTP server may therefore arise. It is recommended to configure only one actively used network interface.

A.3 Parameters for Methods

Unless explicitly specified otherwise for a particular operation, all interface parameters are mandatory. The use of null or empty strings/data structures is therefore not allowed.

A.4 Integration in OSGi

Due to the use of JacORB in ISAAC.lib, problems loading the dependencies may occur. This is because the JacORB classes are not loaded with the default class loader.

In the specific case of OSGi this results in a "java.lang.VerifyError" when calling functions of ISAAC.lib.

A solution to this problem is described in [2] and can be applied to the version of JacORB used by ISAAC.lib. In essence, the approach is based on the fact that, in file MANIFEST.MF of the JacORB jar a dependency on the system.bundle is defined.

Appendix B – Example of a Metadata File

Since the contents of XML files depend on the server's respective data model, the examples given here can not necessarily be applied to every use of the library. They are rather intended to convey an idea of their structure.

B.1 ReadingClient

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

```
<PRODUCT>
```

```
<CARD>
```

```
<dateTimeModified>2009-11-25T13:26:55.19Z</dateTimeModified>
```

```
<identifier>d15701dc-d9bd-11de-b6d4-001999522bc5</identifier>
```

```
<numberOfParts>1</numberOfParts>
```

```
<sourceDateTimeModified>2009-11-25T13:26:55.19Z</sourceDateTimeModified>
```

```
<sourceLibrary>CSD-GER001</sourceLibrary>
```

```
<status>NEW</status>
```

```
</CARD>
```

```
<METADATASECURITY>
```

```
<classification>UNCLASSIFIED</classification>
```

```
<policy>DEU</policy>
```

```
<releasability>XXN</releasability>
```

```
</METADATASECURITY>
```

```
<PART>
```

```
<COMMON>
```

```
<descriptionAbstract> some abstract describing the product </descriptionAbstract>
```

```
<identifierUUID>521a9f52-2309-429d-8c0b-8c95d56c5d79</identifierUUID>
```

```
<type>VIDEO</type>
```

```
</COMMON>
```

```
<COVERAGE>
```

```
<spatialGeographicReferenceBox>
```

```
<lowerRightPoint>
```

```
<latitude>61.0</latitude>
```

```
<longitude>9.6</longitude>
```

```
</lowerRightPoint>
```

```
<upperLeftPoint>
```

```
<latitude>61.1</latitude>
```

```
<longitude>9.5</longitude>
```

```
</upperLeftPoint>
```

```
</spatialGeographicReferenceBox>
```

```
<temporalEnd>2008-02-23T16:05:24.0Z</temporalEnd>
```



```

        <temporalStart>2008-02-23T16:04:00.0Z</temporalStart>
    </COVERAGE>
    <EXPLOITATION_INFO>
        <autoGenerated>false</autoGenerated>
        <level>3</level>
    </EXPLOITATION_INFO>
    <SECURITY>
        <classification>UNCLASSIFIED</classification>
        <policy>DEU</policy>
        <releasability>XXN</releasability>
    </SECURITY>
    <partIdentifier>part1</partIdentifier>
</PART>
<SECURITY>
    <classification>UNCLASSIFIED</classification>
    <policy>DEU</policy>
    <releasability>XXN</releasability>
</SECURITY>
</PRODUCT>

```

B.2 WritingClient

```

<?xml version=" 1.0" encoding=" UTF-8" ?>
<PRODUCT>
    <METADATASECURITY>
        <classification>UNCLASSIFIED</classification>
        <policy>DEU</policy>
        <releasability>XXN</releasability>
    </METADATASECURITY>
    <PART>
        <COMMON>
            <descriptionAbstract>some abstract describing the product</descriptionAbstract>
            <identifierUUID>521a9f52-2309-429d-8c0b-8c95d56c5d79</identifierUUID>
            <type>VIDEO</type>
        </COMMON>
        <COVERAGE>
            <spatialGeographicReferenceBox>
                <lowerRightPoint>
                    <latitude>61.0</latitude>
                    <longitude>9.6</longitude>
                </lowerRightPoint>
                <upperLeftPoint>
                    <latitude>61.1</latitude>
                    <longitude>9.5</longitude>
                </upperLeftPoint>
            </spatialGeographicReferenceBox>
            <temporalEnd>2008-02-23T16:05:24.0Z</temporalEnd>
            <temporalStart>2008-02-23T16:04:00.0Z</temporalStart>

```

```
</COVERAGE>
<EXPLOITATION_INFO>
  <autoGenerated>false</autoGenerated>
  <level>3</level>
</EXPLOITATION_INFO>
<SECURITY>
  <classification>UNCLASSIFIED</classification>
  <policy>DEU</policy>
  <releasability>XXN</releasability>
</SECURITY>
<partIdentifier>part1</partIdentifier>
<VIDEO>
  <category>VIS</category>
  <encodingScheme>MPEG-2</encodingScheme>
</VIDEO>
</PART>
<SECURITY>
  <classification>UNCLASSIFIED</classification>
  <policy>DEU</policy>
  <releasability>XXN</releasability>
</SECURITY>
</PRODUCT>
```

Appendix C – Example of Configuration Files

The configuration files shown here illustrate the structure of ReadingClientProperties or WritingClientProperties objects generated by the PropertyLoader class.

C.1 WritingClient

```
#-----
# connection and general properties
#
# possible manager value = { DataModelMgr, CatalogMgr, ProductMgr, StandingQueryMgr, Crea-
tionMgr, UpdateMgr, OrderMgr }
#-----
server.ior.url=http://localhost:8090/ior
server.manager.CreationMgr.user=myUsername
server.manager.CreationMgr.password=myPassword
server.manager.DataModelMgr.user= myUsername
server.manager.DataModelMgr.password= myPassword
server.manager.UpdateMgr.user= myUsername
server.manager.UpdateMgr.password= myPassword

#-----
# client has a built in http server which is started automatically when
# required, the CSD-Server can retrieve files from and put order results to.
#
# the property describes the port on which the the built in http server is started.
#-----
httpServer.port=9000

#-----
# security values
#
# list of all values = NO
CLASSIFICATION;UNCLASSIFIED;RESTRICTED;CONFIDENTIAL;SECRET;COSMIC TOP SECRET
# to limit the classifications the client will try to send to the server, remove the elements from the
list
#-----
security.classifications=NO CLASSIFICATION;UNCLASSIFIED;RESTRICTED;CONFIDENTIAL;SECRET

#-----
# validation values; both values must be absolute addresses
```

```
#-----
validation.associations.schema=/work/MAJIIIC/Code/CSD-
ThinClients/etc/Product_MAJIIIC_0_1_15.xsd
validation.products.schema=/work/MAJIIIC/Code/CSD-ThinClients/etc/Association_0_1_4.xsd
```

```
#-----
# local system values
#-----
system.name=ClientSystem_1
```

C.2 ReadingClient

```
#-----
# connection and general properties
#
# possible manager value = { DataModelMgr, CatalogMgr, ProductMgr, StandingQueryMgr, Crea-
tionMgr, UpdateMgr, OrderMgr }
#-----
server.ior.url=http://localhost:8090/ior
server.manager.CatalogMgr.user= myUsername
server.manager.CatalogMgr.password= myPassword
server.manager.DataModelMgr.user= myUsername
server.manager.DataModelMgr.password= myPassword

#-----
# client has a built in http server which is started automatically when
# required, the CSD-Server can retrieve files from and put order results to.
#
# the property describes the port on which the built in http server is started.
#-----
httpServer.port=9000

#-----
# local system values
#-----
system.name=ClientSystem_1
```

Appendix D – ISAAC.lib API Javadoc

Each supplied version of the ISAAC.lib software package includes a javadoc software documentation (ISAAC.lib-1.5.0-javadoc.jar), which you can consult for further descriptions of the software.