On the fly

User Guide

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Type of Document** | User Guide | | | | |
| **Reference:** |  | | | | |
| **Issue:** |  | **Revision:** |  | **Status:** | Draft |
| **Created by:** | Gino Mascotti | | | **Date:** | 30/09/2014 |
| **Updated by:** | Gino Mascotti | | |  | 26/05/2015 |
| **Approved by:** |  | | | | |

Summary

**[Project description](#_Toc422216203)** [5](#_Toc422216203)

[Features 5](#_Toc422216204)

[2.0 WSDL Operations 5](#_Toc422216205)

[2.1 WSDL Operations 6](#_Toc422216206)

[REST Operations 7](#_Toc422216207)

[How to retrieve the data 10](#_Toc422216208)

[2.1 OnTheFly 1.0 10](#_Toc422216209)

[2.2 OnTheFly 1.5 11](#_Toc422216210)

[2.3 OnTheFly 2.0 13](#_Toc422216211)

[Presentation 16](#_Toc422216212)

**[Configuration File](#_Toc422216213)** [17](#_Toc422216213)

[Global Settings 17](#_Toc422216214)

[Mapping Settings 19](#_Toc422216215)

[The Store Procedures 21](#_Toc422216216)

[Redirect To RI 25](#_Toc422216217)

[Sample Configuration 27](#_Toc422216218)

**[Error Management](#_Toc422216219)** [31](#_Toc422216219)

[Fault Error 32](#_Toc422216220)

**[Attributes management](#_Toc422216221)** [33](#_Toc422216221)

[Frequency 33](#_Toc422216222)

**[Log Management](#_Toc422216223)** [34](#_Toc422216223)

# **Project description**

The project OnTheFly WebServices was created to execute queries SDMX to different types of databases.  
Is no longer necessary to have the database structured according to a specific model but simply the presence of some store procedures to perform all queries and create all the "On the Fly" required structures.

## Features

OnTheFly WebServices answers all queries Soap or Rest both in SDMX v2.0 that SDMX v2.1  
Implemented funcionalities:

### 2.0 WSDL Operations

Data

|  |  |
| --- | --- |
| WebMethod | Status |
| GetGenericData | Supported |
| GetCompactData | Supported |
| GetUtilityData | Not planned |
| GetCrossSectionalData | Not planned |

Registry-Oriented Web Service Functions to get Structural metadata

|  |  |
| --- | --- |
| WebMethod | Status |
| QueryStructure | Supported |

QueryStructure queriable artefacts

* Dataflow
* Codelist
* ConceptScheme
* CategoryScheme
* KeyFamily
* AgencySchema

### 2.1 WSDL Operations

|  |  |
| --- | --- |
| WebMethod | Status |
| Data |  |
| GetStructureSpecificData | Not planned |
| GetGenericData | Not planned |
| GetStructureSpecificTimeSeriesData | Supported |
| GetGenericTimeSeriesData | Supported |
| Structure usage |  |
| GetDataflow | Supported |
| Structure |  |
| GetDataStructure | Supported |
| Item scheme |  |
| GetCategoryScheme | Supported |
| GetConceptScheme | Supported |
| GetCodelist | Supported |
| GetHierarchicalCodelist | Not planned |
| GetOrganisationScheme | Supported |
| GetReportingTaxonomy | Not planned |
| Other maintainable artefacts |  |
| GetStructureSet | Not planned |
| GetProcess | Not planned |
| GetCategorisation | Supported |
| GetProvisionAgreement | Not planned |
| GetConstraint (content) | Not yet implemented |
| XML Schemas (XSD) |  |
| GetDataSchema | Not planned |
| Generic query for structural metadata |  |
| GetStructures | Not planned |

### REST Operations

Structural Metadata Queries

|  |  |
| --- | --- |
| Resources defined | Status |
| datastructure | Supported |
| categoryscheme | Supported |
| conceptscheme | Supported |
| codelist | Supported |
| hierarchicalcodelist | Not planned |
| organisationscheme | Supported |
| agencyscheme | Supported |
| dataproviderscheme | Not planned |
| dataconsumerscheme | Not planned |
| organisationunitscheme | Not planned |
| dataflow | Supported |
| reportingtaxonomy | Not planned |
| provisionagreement | Not planned |
| structureset | Not planned |
| process | Not planned |
| categorisation | Supported |
| contentconstraint | Not planned |
| attachmentconstraint | Not planned |
| structure | Not planned |

Data Queries

|  |  |  |
| --- | --- | --- |
| Ouput format | Header | Status |
| Generic Data | application/vnd.sdmx.genericdata+xml;version=2.0  application/vnd.sdmx.genericdata+xml;version=2.1 | Supported |
| Structure Specific Data | application/vnd.sdmx.structurespecificdata+xml;version=2.0  application/vnd.sdmx.structurespecificdata+xml;version=2.1 | Supported |
| RDF | application/rdf+xml | Supported |
| DSPL | application/dspl | Supported |
| Json | application/json | Supported |

## How to retrieve the data

Currently the project has three versions:

### 2.1 OnTheFly 1.0

This version uses a single database .STAT to retrieve data and metadata.

You must enter the connectionstring in this database, with configuration parameter ConnectionSTAT

Per tutte le funzionalità verranno sfruttate le storeProcedure del db .STAT

Now let's see in detail how the metadata are gotten:

* **Category and Categorisation**

This metadata can be retrieved by taking information from another database that will have a table "*ContentTree*". This table contains information about the themes. In the CategoryName, in configuration parameter CategorySetting, you can enter the name of the theme that has to be used if there are more than one.

* **Dataflows**

The dataflow is retrieved through the store procedures GetDatasets.  
These will be taken as the the main information to request all the other referenced artifacts.

(.STAT named this StoreProcedure: dbo.proc\_WBS\_GetDatasetList)

* **DataStructures**

The dsd are created on the fly. The application assumes that for each dataset there is one and only one dsd. The code is created with the code of the dataflow and the configuration parameter DsdFormat For each Dsd there is always only one ConceptScheme whose concepts will be taken as described below

* **ConceptScheme**

Also the ConceptScheme are created on the fly, one for each dsd. The code consists of the code of Dataflow with the configuration parameter ConceptSchemeFormat.

The concepts will be so derived:

* + For dimensions using StoreProcedure: GetDimensions

(.STAT named this StoreProcedure: dbo.proc\_WBS\_GetDatasetDimensionsList)

* + For attributes will be read from a file that is in ConfigurationXml\ AttributeConcepts.xml

If you set the configuration parameter: ConceptObservationFlag the attribute FLAG will be added to the DSD

Will be added automatically the OBS\_VALUE primary measures

* **Codelist**

To get the codelist of a dimension, we can derive the constrained version (the list of all the codes that are used through the StoreProcedure: GetDimensionCodelistConstrain.

(.STAT named this StoreProcedure: dbo.proc\_WBS\_GetDimensionMembersListFiltered)

To get the not constrained version you can use two solutions:

* + call the same StoreProcedure for all Dataflow found and merge the results programmatically
  + return an error message by setting the configuration parameter CodelistWhitoutConstrain to false.

About the codelist of the frequency dimension (that is mandatory in case of Time Series dataset) if the codelist is not present into the database it can be located into the Xml file: ConfigurationXml\FrequencyCodelist.xml. In the case the of availability of the codelist into the database it will be taken as any other Dimension.

* **Special Codelist**

Il progetto OnTheFly come per il web service standard RI elabora anche codelist speciali che si dividono in 3 tipologie

* *CL\_TIME\_PERIOD*: restituisce la data iniziale e quella finale del Dataset
* *CL\_COUNT*: restituisce il numero di osservazioni presenti per il Dataset
* *ConstrainMember:* è una richiesta di codelist constrainata non solo a al Dataflow ma anche ad altri CodeItem di altre codelist

Queste sono le uniche informazioni che non venono recuperate attraverso StoreProcedure ma vanno direttamente ad interrogare il database

Queste sono implementate solo per le chiamate specifiche in Soap per Sdmx v2.0

### 2.2 OnTheFly 1.5

In questa versione viene preso in considerazione un solo Database che è .Stat compatibile per il recupero dei metadati e dei dati ma sono state costruite delle StoreProcedure appositamente per il recupero di queste informazioni.

In fase di configurazione questo database sarà chiamato DDB

You must enter the connectionstring in this database, with configuration parameter ConnectionDDB

* **Category and Categorisation**

You can use a StoreProcedure GetCategory that returns directly the necessary information of these artefacts .

(DDB named this StoreProcedure: dbo.proc\_WBS\_GetCategoryAndCategorisation)

* **Dataflows**

The dataflow is retrieved through the store procedures GetDatasets.  
These will be taken as the the main information to request all the other referenced artifacts.

(DDB named this StoreProcedure: dbo.proc\_WBS\_GetDatasetList)

* **DataStructures**

The dsd are created on the fly. The application assumes that for each dataset there is one and only one dsd. The code is created with the code of the dataflow and the configuration parameter DsdFormat For each Dsd there is always only one ConceptScheme whose concepts will be taken as described below

* **ConceptScheme**

Also the ConceptScheme are created on the fly, one for each dsd. The code consists of the code of Dataflow with the configuration parameter ConceptSchemeFormat.

The concepts will be so derived:

* + For dimensions using StoreProcedure: GetDimensions

(DDB named this StoreProcedure: dbo.proc\_WBS\_GetDatasetConceptDimensions)

* + For attributes using StoreProcedure: GetAttributes

(DDB named this StoreProcedure: proc\_WBS\_GetDatasetAttributesList)

If you set the configuration parameter: ConceptObservationFlag the attribute FLAG will be added to the DSD

Will be added automatically the OBS\_VALUE primary measures

* **Codelist**

To get the codelist of a dimension, we can derive the constrained version (the list of all the codes that are used through the StoreProcedure: GetDimensionCodelistConstrain.

(DDB named this StoreProcedure: dbo.proc\_WBS\_GetDimensionCodelistConstrain)

To get the not constrained version you must call the StoreProcedure: GetDimensionCodelistNOConstrain

(DDB named this StoreProcedure: dbo.proc\_WBS\_GetDimensionCodelistNoConstrain)

For codelist concepts of type attribute, we can derive constrained codelist (related to the Dataflow) with the StoreProcedure: GetAttributeCodelistConstrain

(DDB named this StoreProcedure: dbo.proc\_WBS\_GetAttributeCodeListFiltered)

and not constrained using the StoreProcedure: GetAttributeCodelistNOConstrain.

(DDB named this StoreProcedure: dbo.proc\_WBS\_GetAttributeCodeListNoConstrain)

If the codelist are not available into the database it is possible to get the codelist from the Xml file:ConfigurationXml\ AttributeConcepts.xml

About the codelist of the frequency dimension (that is mandatory in case of Time Series dataset) if the codelist is not present into the database it can be located into the Xml file: ConfigurationXml\FrequencyCodelist.xml. In the case the of availability of the codelist into the database it will be taken as any other Dimension.

* **Special Codelist**

This metadata will be retrieved in the same way as the OnTheFly 1.0 version

### 2.3 OnTheFly 2.0

In this version on database is used for data and another one for Metadata (called Mastore). To upgrade to OnTheFly 2.0 version you must enter the connectionstring for Mastore database in configuration parameter MsConnectionString

* **Category and Categorisation**

This metadata will be retrieved in the same way as the OnTheFly 1.5 version

* **Dataflows**

The dataflow is retrieved through the StoreProcedure: GetDataflows

(DDB named this StoreProcedure: dbo.proc\_WBS\_GetDataflowList)

* **DataStructures**

The dsd are recovered effettuando delle interrogazioni dirette nel DB Mastore

* **ConceptScheme**

The ConceptScheme are recovered effettuando delle interrogazioni dirette nel DB Mastore

* **Codelist**

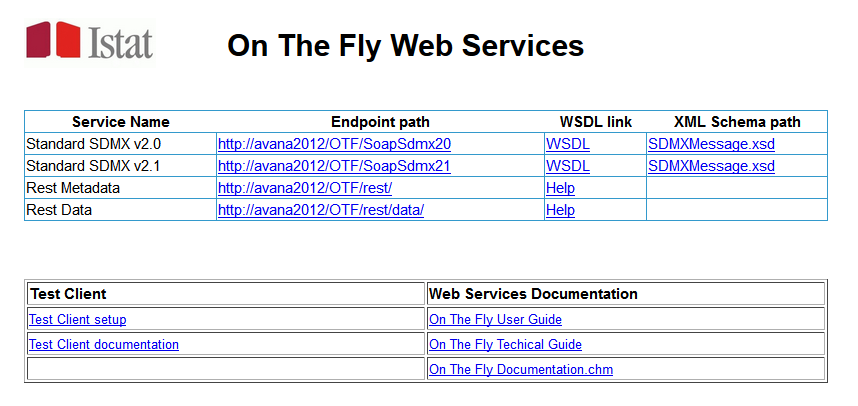
Le Codelist non Constrainate verranno prese effettuando delle interrogazioni dirette nel DB Mastore. The Constrain Codelist will be retrieved in the same way as the OnTheFly 1.5 version

* **Special Codelist**

This metadata will be retrieved in the same way as the OnTheFly 1.5 version

## Presentation

Browsing to where you located the web service it will display the welcome page:



On this page you can view the addresses of the application and download the test client and manuals

# **Configuration File**

The configuration file of the project OnTheFly WebServices:

**bin\ServiceConfiguration.xml**

This file is composed of two sections

* **Global Setting**
* **Mapping Settings**

## Global Settings

In this section you will find all the configuration parameters required and used in all versions of the OnTheFly  
The parameters are composed of

<add key="[key]" value="[value]" />

|  |  |
| --- | --- |
| Key | Description |
| MainAgencyId | AgencyScheme Structure and descriptions  It is possible insert in this tag one or more name with this syntax:  <Name LocaleIsoCode="en"><![CDATA[Agency Name]]></Name> |
| OrganisationScheme | OrganisationScheme Structure and descriptions  It is possible insert in this tag one or more name with this syntax:  <Name LocaleIsoCode="en"><![CDATA[ISTAT AGENCIES]]></Name> |
| Version | Default Version of artefacts |
| DatasetTitle | It can be set to true/false value and determinates if you want to show a Dataset Title in data query result at Dataset Level. It will take the first description available |
| UserName | This parameter is required by some Store procedure |
| Domain | This parameter is required by some Store procedure |
| LogLocation | It determinates a location of application Log. It is possible to use a special Word as %Temp% to identify a temporary Path |
| LogLevel | Possible value:  **"None"** -> No log  **"All"** -> All log (Debug Mode)  **"Warning**" -> Only Warning and Error Log  **"Error"** -> Only Error Log |
| HeaderSettings | It is possible to configure all the Header Settings that you want to return |

## Mapping Settings

In this section you can save one or more possible configurations, depending on the version you want to use.

In the main node MappingSettings there is an attribute UsedSettings that should be enhanced with the name of the configuration you want to use

To create a new configuration you must add:

<MappingSetting id="SettingsId">

The *id* is the name you want to give to the configuration.

Possible parameters:

|  |  |
| --- | --- |
| Key | Description |
| ConnectionSTAT | ConnectionString for connection to SqlServer Database .STAT  In this Database there must be present all Store Procedure described below |
| ConnectionStringDDB | ConnectionString for connection to SqlServer Database .STAT compatible (DDB)  In this Database there must be present all Store Procedure described below  If this parameter is valorized the version of the program automatically change in 1.5 |
| MsConnectionString | ConnectionString for connection to MappingStore Database  If this parameter is valorized the version of the program automatically change in 2.0 |
| CodelistWhitoutConstrain | Set to true for merging codelist of all Dataflow (low performance)  in case of receiving Query Codelist whitout Constrain (Dataflow reference)  otherwise an exception will be generated. |
| ConceptObservationFlag | You can put the FLAG concept in dsd, in the value field you must specify the ID in the *attachmentLevel* (the level where the data should be inserted inside), *assignmentStatus* if it is mandatory or optional and descriptions names.  The codelist of ConceptObservationFlag will be taken to the store procedure “GetFlags”.  If you do not want to add this attribute, you must remove the tag from the configuration file. |

|  |  |
| --- | --- |
| CategorySettings |  |
| ConnectionStringCategory | ConnectionString Alternative to retrieve CategoryScheme, Category, Categorisation  In this Database must be present a table ContentTree |
| CategoryName | In this parameter you can set the Category name that you want to use in ContentTree Table |

The formats of the id of the artifacts made

|  |  |
| --- | --- |
| Key | Description |
| DsdFormat | Required by KeyFamily and DataStructure ID. The characters {0} will be sobstituted with Dataflow ID |
| ConceptSchemeFormat | Required by ConceptScheme ID. The characters {0} will be sobstituted with Dataflow ID |
| CodelistFormat | Required by Codelist ID. The characters {0} will be sobstituted with Concept ID |
| CategorySchemeFormat | Required by CategoryScheme ID. The characters {0} will be sobstituted with Dataflow ID |
| CategorisationFormat | Required by Categorisation ID. The characters {0} will be sobstituted with Dataflow ID |

### The Store Procedures

|  |  |
| --- | --- |
| Key | Description |
| GetCategory | Get Category and Categorisation, if this store procedure does not exist and the parameter *CategorySettings* is populated, the OnTheFly system will get Category information from other Database  *Parameter:*  UserName: User name  Domain: user domain  TimeStamp: date to require data from last update |
| GetDatasets | Get Datasets, required in OTF 1.0  *Parameter:*  UserName: User name  Domain: user domain  TimeStamp: date to require data from last update |
| GetDimensions | Get Dimensions,required in OTF 1.0  *Parameter:*  Code: Dataset Code  TimeStamp: date to require data from last update |
| GetDimensionCodelistConstrain | Get the constrained codelist of a dimension (it requires the specification of the dataset), required in OTF 1.0  *Parameter:*  DatasetCode: Dataset Code  DimCode: Dimension Code  UserName: User name  Domain: user domain  TimeStamp: date to require data from last update |
| GetDimensionCodelistNOConstrain | Get the a not constrained codelist of a dimension, if the store procedure does not exist, the OnTheFly system will check the flag “CodelistWhitoutConstrain”  *Parameter:*  DimCode: Dimension Code  Depth: depth of the level of codelist  UserName: User name  Domain: user domain  TimeStamp: date to require data from last update |
| GetAttributes | Get Attributes, if the store procedure does not exist, the OnTheFly system wil get a Attribute from File “Attributes.xml”  *Parameter:*  Code: Dataset Code  TimeStamp: date to require data from last update |
| GetAttributeCodelistConstrain | Get the constrained codelist of an attribute (only used in Dataset specified), if this store procedure does not exist, the OnTheFly system will get an Attribute from File “Attributes.xml”  *Parameter:*  DatasetCode: Dataset Code  AttributeCode: Attribute Code  Depth: depth of the level of codelist  UserName: user name  Domain: user domain  TimeStamp: date to require data from last update |
| GetAttributeCodelistNOConstrain | Get the not constrained codelist of an attribute, if this store procedure does not exist, the OnTheFly system will get an Attribute from File “Attributes.xml”  *Parameter:*  AttributeCode: Attribute Code  Depth: depth of the level of codelist  UserName: user name  Domain: user domain  TimeStamp: date to require data from last update |
| GetData | Get Data, required  *Parameter:*  DatasetCode: Dataset Code  WhereStatement: where condition each parameter in this format: [$parameter$].value  Time: where condition for time parameter  DataSetAction: Dataset action  UserName: User nameDomain: user domain  TimeStamp: date to require data from last update |
| GetGroups | Get Groups, used in OnTheFly v2.0  *Parameter:*  DatasetCode: Dataset Code  Columns: each required item separated by Comma  UserName: user name  Domain: user domain  TimeStamp: date to require data from last update |
| GetDataflows | Get Dataflows, used in OnTheFly v2.0  *Parameter:*  UserName: user name  Domain: user domain  TimeStamp: date to require data from last update |

### Redirect To RI

Per la versione OnTheFly 2.0 c’è la possibilità di effettuare una richiesta preventiva al Database Mastore e controllare se I dati di un determinato Dataflow sono presenti (mappati) nel Database DDB oppure stanno su un altro Database. In questo caso inserendo nella configurazione un WebService RI Standard che punta al medesimo Mastore le richieste di dati e Metadati constrainati vengono rigirati direttamente a questo costruendo delle query sdmx che verranno spedite via SOAP.

Per la poter usufruire di questa funzionalità è necessario configurare I 3 endpoint del WebService RI Standard in questo modo:

<RIWebService>

<WebServicesUrl EndPoint=<http://../NSIStdV20Service> EndPointType="V20"></WebServicesUrl>

<WebServicesUrl EndPoint=<http://../SdmxService> EndPointType="V21"></WebServicesUrl>

<WebServicesUrl EndPoint=<http://../rest> EndPointType="REST"></WebServicesUrl>

</RIWebService>

### Sample Configuration

**OnTheFly 1.0**

<MappingSetting id="OnTheFly 1.0">

<add key="ConnectionSTAT" value="Data Source=xxx;Initial Catalog=xxx;

User ID=xxx;Password=xxx;"/>

<CategorySettings>

<add key="ConnectionStringCategory" value="Data Source=xxx;Initial Catalog=xxx;

User ID=xxx;Password=xxx;"/>

<add key="CategoryName" value="I-Stat-Dati"/>

</CategorySettings>

<!--For this format don't remove symbol {0}-->

<add key="DsdFormat" value="{0}\_DSD"/>

<add key="ConceptSchemeFormat" value="{0}\_CS"/>

<add key="CategorySchemeFormat" value="{0}\_CategoryScheme"/>

<add key="CodelistFormat" value="CL\_{0}"/>

<add key="CategorisationFormat" value="{0}\_Categorisation"/>

<add key="CodelistWhitoutConstrain" value="true"/>

<add key="ConceptObservationFlag" value="OBS\_STATUS" attachmentLevel="Observation"

assignmentStatus="Conditional">

<Name LocaleIsoCode="en"><![CDATA[Observation status]]></Name>

<Name LocaleIsoCode="fr"><![CDATA[Observation status]]></Name>

</add>

<StoreProcedureSettings>

<store key ="GetDatasets" name="dbo.proc\_WBS\_GetDatasetList"/>

<store key ="GetDimensions" name="dbo.proc\_WBS\_GetDatasetDimensionsList"/>

<store key ="GetDimensionCodelistConstrain"

name="dbo.proc\_WBS\_GetDimensionMembersListFiltered"/>

<store key ="GetFlags" name="dbo.proc\_WBS\_GetDataSetFlags"/>

<store key ="GetData" name="dbo.proc\_SDMX\_GetObservations"/>

</StoreProcedureSettings>

</MappingSetting>

**OnTheFly 1.5**

<MappingSetting id="OnTheFly 1.5">

<add key="ConnectionDDB" value="Data Source=xxx;Initial Catalog=xxx;

User ID=xxx;Password=xxx;"/>

<!--For this format don't remove symbol {0}-->

<add key="DsdFormat" value="{0}\_DSD"/>

<add key="ConceptSchemeFormat" value="{0}\_CS"/>

<add key="CategorySchemeFormat" value="{0}\_CategoryScheme"/>

<add key="CodelistFormat" value="CL\_{0}"/>

<add key="CategorisationFormat" value="{0}\_Categorisation"/>

<add key="CodelistWhitoutConstrain" value="true"/>

<StoreProcedureSettings>

<store key ="GetDatasets" name="dbo.proc\_WBS\_GetDatasetList"/>

<store key ="GetCategory" name="dbo.proc\_WBS\_GetCategoryAndCategorisation"/>

<store key ="GetData" name="dbo.proc\_SDMX\_GetObservations"/>

<store key ="GetDataflows" name="dbo.proc\_WBS\_GetDataflowList"/>

<store key ="GetAttributes" name="dbo.proc\_WBS\_GetDatasetAttributesList"/>

<store key ="GetAttributeCodelistNOConstrain"

name="dbo.proc\_WBS\_GetAttributeCodeListNoConstrain"/>

<store key ="GetAttributeCodelistConstrain"

name="dbo.proc\_WBS\_GetAttributeCodeListFiltered"/>

<store key ="GetDimensions" name="dbo.proc\_WBS\_GetDatasetConceptDimensions"/>

<store key ="GetDimensionCodelistNOConstrain"

name="dbo.proc\_WBS\_GetDimensionCodelistNoConstrain"/>

<store key ="GetDimensionCodelistConstrain"

name="dbo.proc\_WBS\_GetDimensionCodelistConstrain"/>

</StoreProcedureSettings>

</MappingSetting>

**OnTheFly 2.0**

<MappingSetting id="OnTheFly 1.5">

<add key="ConnectionDDB" value="Data Source=xxx;Initial Catalog=xxx;

User ID=xxx;Password=xxx;"/>

<add key=" MsConnectionString " value="Data Source=xxx;Initial Catalog=xxx;

User ID=xxx;Password=xxx;"/>

<!--For this format don't remove symbol {0}-->

<add key="CategorySchemeFormat" value="{0}\_CategoryScheme"/>

<add key="CategorisationFormat" value="{0}\_Categorisation"/>

<StoreProcedureSettings>

<store key ="GetGroups" name="dbo.proc\_SDMX\_GetGroups"/>

<store key ="GetData" name="dbo.proc\_SDMX\_GetObservations"/>

<store key ="GetDataflows" name="dbo.proc\_WBS\_GetDataflowList"/>

<store key ="GetAttributeCodelistConstrain"

name="dbo.proc\_WBS\_GetAttributeCodeListFiltered"/>

<store key ="GetDimensionCodelistConstrain"

name="dbo.proc\_WBS\_GetDimensionCodelistConstrain"/>

<store key ="MSGetCategoryAndCategorisation"

name="dbo.proc\_WBS\_GetCategoryAndCategorisation"/>

</StoreProcedureSettings>

<RIWebService>

<WebServicesUrl EndPoint=<http://avana2012.pc.istat.it/WS_FEB2015/NSIStdV20Service>

EndPointType="V20"></WebServicesUrl>

<WebServicesUrl EndPoint=<http://avana2012.pc.istat.it/WS_FEB2015/SdmxService>

EndPointType="V21"></WebServicesUrl>

<WebServicesUrl EndPoint=<http://avana2012.pc.istat.it/WS_FEB2015/rest>

EndPointType="REST"></WebServicesUrl>

</RIWebService>

</MappingSetting>

# **Error Management**

The project OnTheFly, in case of error, returns an SDMX error message for both errors, resulting from calls Soap and for those calls from Rest.

**<s:Fault**>

<**faultcode**>s:500<**/faultcode**>

**<faultstring xml:lang**="it-IT">Internal Server Error<**/faultstring**>

<**faultactor**>GetGenericData<**/faultactor**>

<**detail**>

**<Error Type**="InternalError">

<**Message**>"WS Internal Error<**/Message**>

<**MessageDetail**>Dsd Not found for Dataflow code: TestDataFLOW<**/MessageDetail**>

<**Source**>FlyEngine.Model.RetrievalManager<**/Source**>

<**/Error**>

<**/detail**>

<**/s:Fault**>

The errors can be divided into two types:

* "Derived from the project"

They are all design errors, configuration or data retrieval from the database

* "Derived from CommonApi"  
  These are all errors that are triggered by commonApi as parse query or immutable instanced of the Sdmx Objects

In either case the system will return:  
*FaultCode* and *faultString* (all possible configurations are described below)  
*faultactor*, describes the entrypoint of the application that was called: detail*Error*: this is the type attribute that indicates whether the error is "Derived from the project" or "Derived from CommonApi"  
*Message* and *MessageDetail* are descriptions of the error  
*Source* is the name of the source object that triggered the error

For all errors "derivatives from the project" you can change the description and *FaultCode* editing the xml file that is located in

**“Bin/ConfigurationXml/ErrorDescription.xml”**

## Fault Error

|  |  |  |
| --- | --- | --- |
| Code | SdmxError | Description |
| 100 | NoResultsFound | No Results Found |
| 110 | Unauthorised | Unauthorised |
| 130 | ResponseTooLarge | Response Too Large |
| 140 | SyntaxError | Syntax Error |
| 150 | SemanticError | Semantic Error |
| 500 | InternalServerError | Internal Server Error |
| 501 | NotImplemented | Not Implemented |
| 503 | ServiceUnavailable | Service Unavailable |
| 510 | ResponseSizeExceedsServiceLimit | Response Size Exceeds Service Limit |

More information you can find in the manual SDMX WebServicesGuidelines

# **Attributes management**

The project OnTheFly has A supported feature that allows you to insert the attributes and their codelist even without of StoreProcedure dealing retrieve this information.

The system will retrieve the attributes and their codelist from a file that is located in

**“Bin/ConfigurationXml/AttributeConcepts.xml”**

In this file you can assign each dataflow attributes to their codelist in this way:

<Attribute Code="AVAILABILITY" attachmentLevel="Dataset" assignmentStatus="Mandatory">

<Name LocaleIsoCode="en"><![CDATA[Attribute AVAILABILITY]]></Name>

<Name LocaleIsoCode="fr"><![CDATA[Attribute AVAILABILITY]]></Name>

<Codelist>

<Code value="0">

<Name LocaleIsoCode="en"><![CDATA[Available]]></Name>

<Name LocaleIsoCode="fr"><![CDATA[Available]]></Name>

</Code>

<Code value="1">

<Name LocaleIsoCode="en"><![CDATA[not Available]]></Name>

<Name LocaleIsoCode="fr"><![CDATA[not Available]]></Name>

</Code>

</Codelist>

</Attribute>

How you can see, we have defined an attribute by Code and Name nodes.

Required xml attributes:

1. attachmentLevel describes where they will be placed within the attributes of the data structure and can be:
   * Dataset
   * Observation
   * DimensionGroup (at the level of the series)
2. And assignmentStatus describes whether the attribute is Mandatory or Conditional

In node Codelist you can assign the codelist with the respective Code with their value and Name which can also be nested, in this case the system will automatically assign them the ParentCode

## Frequency

The dimension of type *frequency* is required by the system OnTheFly, since all data requests are type TimeSeries. But it is still possible that in a given dataflow this dimension is not present. In this case the system will automatically add the Frequency Dimension and the related codelist picking her recover from the file that is located in:

**“Bin/ConfigurationXml/FrequencyCodelist.xml”**

# **Log Management**

In the OnTheFly system there is a part of the configuration dedicated to the log.

|  |  |
| --- | --- |
| LogLocation | Determinate a location of application Log. Is possible use a special Word as %Temp% for identify a temporary Path |
| LogLevel | Possible value:  **"None"** -> No log  **"All"** -> All log (Debug Mode)  **"Warning**" -> Only Warning and Error Log  **"Error"** -> Only Error Log |

If the configuration is properly set: *LogLevel* not *None* and *LogLocation* is an existing writable (the web service has to allowed to write in it) folder, the system will write a log file every hour (if operations are performed).