ESCWA SDMX Workshop

Session: Web Services
SDMX REST Interface
RESTful web services

These slides are based on the presentation given at the SDMX Global Conference 2011 by the ECB
What are web services?

- Make resources published on the Internet available via an API
- XML used as representation
- Application-to-application interactions
- Main benefits:
  - Automated processes
  - Language- and OS-independent
  - Standardised protocols
  - Decentralised architecture with loosely-coupled components
SDMX RESTful web services: Basic principles

• Based on REST principles:
  – Resource addressability (URI)
  – Resource manipulation (HTTP GET, POST, etc)
  – Resource representation (HTTP content negotiation)

• Distinction between:
  – Parameters that identify a resource (in the path)

• Parameters that further refine the results (in the query string)
Structural metadata queries:
Requirements

• Retrieve any maintainable artefact
• Resolve direct references
• Resolve artefacts using the matching artefact
• Retrieve minimal version of the artefact
Structural metadata queries:
Parameters

• Parameters used for identifying a resource: id, agency id and version (in the path)
  – Keywords can be used (ALL and LATEST)
• Parameters used to further refine the results: references and detail (in the query string)
• Syntax:
  protocol://ws-entry-point/resource/agencyID/resourceID/version[/?references=String&detail=String]
Structural metadata queries: Examples

- Retrieve version 1.1 of the DSD MDG maintained by IAEG
  [http://ws-entry-point/datastructure/IAEG/MDG/1.1](http://ws-entry-point/datastructure/IAEG/MDG/1.1)
- Retrieve the latest version in production of the DSD MDG maintained by IAEG, as well as the code lists and the concepts used in the data structure:
- Retrieve all DSDs maintained by the IAEG, as well as the dataflows using these DSDs:
  [http://ws-entry-point/datastructure/IAEG?references=dataflow](http://ws-entry-point/datastructure/IAEG?references=dataflow)
- Retrieve all the artefacts maintained by the IAEG, but without the details:
  [http://ws-entry-point/structure/IAEG?detail=allstubs](http://ws-entry-point/structure/IAEG?detail=allstubs)
Data queries: Requirements

• Retrieve statistical data and metadata using keys (with wildcarding and OR operator), flows and providers.
• Refine queries, using time information (start period and end period).
• Retrieve updates and revisions only.
Data queries: Parameters

- Parameters used for identifying a resource: flow id, key and provider id (in the path)
  - Keywords can be used (ALL)
- Parameters used to further refine the results: startPeriod, endPeriod, updatedAfter, firstNObservations, lastNObservations and dimensionAtObservation, detail

Syntax:

protocol:// ws-entry-point/resource/flow/key/provider[/?startPeriod=StandardTimePeriodType&endPeriod=StandardTimePeriodType&updatedAfter=timestamp&firstNObservations=uint&firstNObservations=uint&dimensionAtObservation=string&detail=String]
Data queries: Examples

• Retrieve the data and attributes for the series A. SN_ITK_DEFC.POP.T. 000_099_Y.T.GHA.S supplied by the IAEG for the MDG_STATS_WEB dataflow: http://ws-entry-point/Data/MDG_STATS_WEB/A.SN_ITK_DEFC.POP.T. 000_099_Y.T.GHA.S/IAEG

• Retrieve the data (and not the attributes), provided by the IAEG for the MDG_STATS_WEB dataflow, for the supplied series keys, using wildcarding for the second dimension: http://ws-entry-point/Data/MDG_STATS_WEB/A..POP.T.000_099_Y.T.GHA.S/IAEG?detail=dataonly

• Retrieve the data matching the supplied series keys (using the OR operator) and restricting the start and end dates: http://ws-entry-point/Data/MDG_STATS_WEB/A..POP.T.000_099_Y.T.GHA+CHL.S/?startPeriod=2009-05-01&endPeriod=2009-05-31
Resource representation & HTTP content negotiation

- Selection of the appropriate format using HTTP Content Negotiation (Accept HTTP Header)
- Syntax: application/vnd.sdmx.[format]+xml;version=[version]
  Examples:
  - application/vnd.sdmx.generic+xml;version=1.0
  - application/vnd.sdmx.structure+xml;version=2.1
- In case the client does not specify the desired format:
  - Returns the most recent version of the SDMX-ML Structure format for structural metadata queries;
  - Returns the most recent version of the SDMX-ML Generic Data format for data queries;
  - Returns the most recent version of the SDMX-ML Generic Metadata format for metadata queries.
Handling errors: HTTP status codes (I)

• Indicate errors using the proper HTTP status code
• Whenever appropriate, SDMX-ML error message can be used
• 3 error categories:
  – 000 – 499: Error generated from client
  – 500 – 999: Error generated from server
  – 1000+: Service custom messages
## Handling errors: HTTP status codes (II)

<table>
<thead>
<tr>
<th>SDMX error</th>
<th>HTTP status code</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 - No results found</td>
<td>404</td>
</tr>
<tr>
<td>110 - Unauthorized</td>
<td>401</td>
</tr>
<tr>
<td>130 - Response too large due to client request</td>
<td>413</td>
</tr>
<tr>
<td>140 - Syntax error</td>
<td>400</td>
</tr>
<tr>
<td>150 - Semantic error</td>
<td>400</td>
</tr>
<tr>
<td>500 - Internal Server error</td>
<td>500</td>
</tr>
<tr>
<td>501 - Not implemented</td>
<td>501</td>
</tr>
<tr>
<td>503 - Service unavailable</td>
<td>503</td>
</tr>
<tr>
<td>510 - Response size exceeds service limit</td>
<td>413</td>
</tr>
<tr>
<td>1000+</td>
<td>500</td>
</tr>
</tbody>
</table>