SDMX
GUIDELINES
FOR THE
USE OF WEB SERVICES

(VERSION 2.0)

November 2005

1 INTRODUCTION

Web services represent the coming generation of Internet technologies. They allow computer applications to exchange data directly over the Internet, essentially allowing modular or distributed computing in a more flexible fashion than ever before. In order to allow web services to function, however, many standards are required: for requesting and supplying data; for expressing the enveloping data which is used to package exchanged data; for describing web services to one another, to allow for easy integration into applications that use other web services as data resources.

SDMX, with its focus on the exchange of data using Internet technologies, will provide some of these standards as regards statistical data and metadata. Many web-services standards already exist, however, and there is no need to re-invent them for use specifically within the statistical community. Specifically, SOAP (which originally stood for the “Simple Object Access Protocol”) and the Web Services Description Language (WSDL) can be used by SDMX to complement the data and metadata exchange formats they are standardizing.

Despite the promise of SOAP and WSDL, it has been discovered that various implementations by vendors were not, in fact, interoperable. It was for this reason that the Web Services - Interoperability (WS-I) initiative was started. This consists of a group of vendors who have all implemented the same web-services standards the same way, and have verified this fact by doing interoperability tests. They publish profiles describing how to use web services standards interoperably. SDMX uses the work of WS-I as appropriate to meet the needs of the statistical community.

This document is not normative – it intends to suggest a best practice in using SDMX-ML documents and web services standards for the exchange of statistical data and metadata. In future, it is anticipated that normative standards for the use of web-services technologies may be offered by the SDMX Initiative, based on the guidelines provided here.

2 WEB SERVICES AND SDMX-ML

Conventional applications and services traditionally expose their functionality through application programming interfaces (APIs). Web services are no different – they provide a public version of the function calls which can be accessed over the web using web-services protocols. In order to make a set of web services interoperate, it is necessary to have a standard abstraction, or model, on which these public functions are based. SDMX benefits from having a common information model, and it is a natural extension to use the SDMX Information Model as the basis for standard web-services function calls.

Web services exchange data in an XML format: this is how the data passed between web services is formatted. SDMX-ML, as a standard XML for exchanging data and structural metadata within the statistical realm, provides a useful XML format for the public serialization of web-services data. While there are some techniques for simple web-services data exchanges – remote procedure calls (RPCs) – which are often used, the use of a set of XML exchanges based on a common information model is seen as a better approach for achieving interoperability.

There are several different document types available within SDMX-ML, and all are potentially important to the creators and users of SDMX web services.

1. **The "Envelope" Message:** This is for use in non-web-services applications, as it is partially redundant with SOAP. All SDMX messages can be used without this wrapper.

2. **The "Structure" Message:** This message describes the concepts, key families, and codelists which define the structure of statistical data. Every SDMX-compliant data set must have a key family structure described for it. This XML description must be available from an SDMX web service when it is asked for. It also contains structural metadata used for the exchange of Reference Metadata.

3. **The "Generic" Data Message:** This is the "generic" way of marking up SDMX data. This schema describes a non-key-family-specific format for exchanging SDMX data, and it is a requirement that every SDMX web service make its data available in at least this form. (Often, the other key-family-specific XML forms for expressing data will also be supported in parallel services).

4. **The "Compact" Data Message:** This is a standard schema format derived from the structure description using a standardized mapping, and many standard tags. It is specific to the structure of a particular key family, and so every key family will have its own "Compact" schema. It is designed to enable the transfer of large data sets, and to permit incremental updates. This is a data format that a web service may wish to provide, depending on the requirements of the data they exchange.

5. **The "Utility" Data Message:** This is probably of less interest to those providing SDMX web services, but may be useful in some domains. Like the "Compact" data message, it is specific to the key-family of the data it is used to mark up. It is derived according to standard mappings from the key-family description. It is designed to provide a typical XML schema for a particular type of statistical data, as used by many common XML editing and presentation tools. Unlike the Compact Message, this data is quite verbose, and requires a complete data set. Consequently, it cannot be used for incremental updates.

6. **The "Cross-Sectional Data"** Message: This message allows for more than a single observation to be supplied with a given observation time value, and further allows some values of the key to be specified at the observation level (instead of at the series level or above, like time-series-related SDMX data formats). This is particularly useful for some statistical data sets. Like the Compact message and the Utility message, it is derived from the structure description according to standard mappings.

7. **The "Query" Message:** This is the message used to invoke an SDMX web service. It is generic across all key families and reference metadata structural defintions, but makes its queries in terms of the values specified for the concepts of a specific structure (as specified in a structure description). It allows users to query for data, concepts, codelists, key families, and metadata structure definitions - these functions should thus all be supported by an SDMX web service (depending on whether support is provided for data queries and/or metadata queries.)

8. **The "RegistryInterfaces" Message:** All of the Registry Interfaces are sub-elements of this SDMX-ML Message type. They are more fully described in the SDMX Registry Specification.

9. **The "Reference Metadata" Message:** This is a message used to report reference metadata concepts, which is generic across all types of reference metadata structural descriptions.

10. **The "MetadataReport" Message:** This is a message used to report reference metadata concepts specific to a particular metadata structure definition.

Note that for each data message, a global element is available for use with SOAP envelopes. SDMX web services should not use the `<wsdl:types>` element, but instead use the `<wsdl:import>` element to specify the schemas concerned.

Note that all SDMX web services are required to support the exchanges which enable querying on key families, codelists, and concepts, and it is recommended that they support at minimum the Generic Data format. This guarantees that at least one data format will exist in common between the data publisher and any user of the web service. In many cases, the more optimized data formats will be more commonly used and requested, as they are optimized for use with the processes commonly associated with that data. Guaranteeing a single, common data format is, however, the basis on which widespread interoperability can be built for future uses of the data.

3 EXCHANGE PATTERNS FOR SDMX WEB SERVICES

All SDMX web services should be described using WSDL instances, according to the use of WSDL to specify the aspects of this multiple-message exchange which they support. The global element for each XML data and metadata format within SDMX should be specified as the content of the replies to each exchange. The function names for each identified pattern are specified below, along with the type of SDMX-ML payload.

Because SOAP RPC is not supported, the "parameters" of each function are simply an instance of the appropriate SDMX-ML message type. As noted above, `<wsdl:import>` should be used to specify the schema for a multiple-message exchange.

3.1 Data- and Metadata-Oriented Web Service Functions

Because SDMX offers a number of data formats (although it only requires one), and because it concerns itself both with data and with the structural metadata often needed to understand and process that data, the SDMX web service is composed of a set of data exchanges. Thus, the SDMX web service implements a "multiple-message exchange pattern" (in WSDL terminology). These exchanges are enumerated below, along with an indication of whether the SDMX web service is required to support them:

1. **Obtain Key Family:** This is an exchange invoked by the Query Message, for which the return message is a key family description or descriptions, expressed as a Structure Message. Support is recommended if data queries are supported. The function should be called “GetKeyFamily(Query)” with an input Query Message and a response Structure Message carrying a valid instance of the KeyFamilies element.

2. **Obtain Codelists:** This is an exchange invoked by the Query Message, for which the return is a codelist or codelists, expressed as a Structure Message. Support is recommended. The function should be called “GetCodelists(Query)” with an input Query Message and a response Structure Messag carrying a valid instance of the Codelists element.

3. **Obtain Concepts:** This is an exchange invoked by the Query message, for which the response is a concept scheme or concept schemes, expressed as a Structure Message. Support is recommended. The function should be called “GetConcepts(Query)” with an input Query Message and a response Structure Message carrying a valid instance of the Concepts element or ConceptSchemes element.

4. **Obtain Metadata Structure Definition:** This is an exchange invoked by the Query Message, for which the response is a metadata structure definition, expressed as a Structure Message. Support is recommended if metadata queries are supported. The function should be called “GetMetadataStructure(Query)” with an input Query Message and a response Structure Message carrying a valid MetadataStructureDefinitions element.

5. **Obtain Generic Data:** This is an exchange invoked by the Query Message, for which the response is data marked up according to the Generic Data Message. Support is recommended. The function should be called “GetGenericData(Query)”.

6. **Obtain Compact Data:** This is an exchange invoked by the Query Message, for which the response is data marked up according to the Compact Data Message. The function should be called “GetCompactData(Query)”.

7. **Obtain Utility Data:** This is an exchange invoked by the Query Message, for which the response is data marked up according to the Utility Data Message. The function should be called “GetUtilityData(Query)”.

8. **Obtain Cross-Sectional Data:** This is an exchange invoked by the Query Message, for which the response is data marked up according to the Cross-Sectional Data Message. The function should be called “GetCrossSectionalData(Query)”.

9. **Obtain Reference Metadata:** This is an exchange invoked by the Query Message, for which the response is reference metadata marked up according to the Reference Metadata Message. The function should be called “GetReferenceMetadata(Query)”.

10. **Obtain Metadata Report:** This is an exchange invoked by the Query Message, for which the response is reference metadata marked up according to the

Metadata Report Message. The function should be called "GetMetadataReport(Query)".

11. **Obtain Hierarchical Codelist:** This is an exchange invoked by the Query Message, for which the return is a hierarchical codelist or hierarchical codelists, expressed as a Structure Message. Support is optional. The function should be called "GetHierarchicalCodelists(Query)" with an input Query Message and a response Structure Message carrying a valid instance of the HierarchicalCodelists element.

12. **Obtain Structure Set:** This is an exchange invoked by the Query Message, for which the return is a structure set or structure sets, expressed as a Structure Message. Support is optional. The function should be called "GetStructureSets(Query)" with an input Query Message and a response Structure Message carrying a valid instance of the StructureSets element.

13. **Obtain Reporting Taxonomy:** This is an exchange invoked by the Query Message, for which the return is a reporting taxonomy or reporting taxonomies, expressed as a Structure Message. Support is optional. The function should be called "GetReportingTaxonomies(Query)" with an input Query Message and a response Structure Message carrying a valid instance of the ReportingTaxonomies element.

14. **Obtain Process:** This is an exchange invoked by the Query Message, for which the return is a process or processes, expressed as a Structure Message. Support is optional. The function should be called "GetProcesses(Query)" with an input Query Message and a response Structure Message carrying a valid instance of the Processes element.

3.2 Registry-Oriented Web Service Functions

1. **Submit Subscription to SDMX Registry/Repository:** This is an exchange invoked by the SubmitSubscriptionRequest message, for which the response is a confirmation in the form of a SubmitSubscriptionResponse message. The function should be called "SubmitSubscription(SubmitSubscriptionRequest)".

2. **Submit Registration of Data or Reference Metadata Sets to Registry:** This is an exchange invoked by the SubmitRegistrationRequest message, for which the response is a confirmation in the form of a SubmitRegistrationResponse message. The function should be called "SubmitRegistration(SubmitRegistrationRequest)".

3. **Query Data or Reference Metadata Registry:** This is an exchange invoked by the QueryRegistrationRequest message, for which the response is a confirmation in the form of a QueryRegistrationResponse message. The function should be called "QueryRegistration(QueryRegistrationRequest)."

4. **Submit Structural Metadata to Repository:** This is an exchange invoked by the SubmitStructureRequest message, for which the response is a confirmation in the

form of a SubmitStructureResponse message. The function should be called "SubmitStructure(SubmitStructureRequest)."

5. **Query Structural Metadata in Repository:** This is an exchange invoked by the QueryStructureRequest message, for which the response is a confirmation in the form of a QueryStructureResponse message. The function should be called "QueryStructure(QueryStructureRequest)."

6. **Submit Provisioning Metadata to Repository:** This is an exchange invoked by the SubmitProvisioningRequest message, for which the response is a confirmation in the form of a SubmitProvisioningResponse message. The function should be called "SubmitProvisioning(SubmitProvisioningRequest)."

7. **Query Provisioning Metadata in Repository:** This is an exchange invoked by the QueryProvisioningRequest message, for which the response is a confirmation in the form of a QueryProvisioningResponse message. The function should be called "QueryProvisioning(QueryProvisioningRequest)."

4 COMPLIANCE WITH WS-I

To ensure interoperability between SDMX web services, compliance with sections of the WS-I Profile 1.1 is recommended for all SDMX web services. The documentation can be found at http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html. The recommended sections are those concerning the use of SOAP and WSDL. UDDI, while useful for advertising the existence of SDMX web services, is not necessarily central to SDMX interoperability.

5 LARGE DATA AND METADATA SETS AND QUERYING

Because some queries may produce huge numbers of data points or large amounts of reference metadata as a response, it is recommended that an SDMX web service support the use of the "DefaultLimit" field in the SDMXQuery message. If a response will be larger than the suggested default limit in the query, then the response should be truncated. A truncated response is a partial response, but must still be a valid SDMX-ML document. The fact that it is truncated should be indicted with the "Truncated" field in the Header element of the response message.

Note that the default limit is to be interpreted as an order-of-magnitude suggestion, and not as a literal limit – it is not always easy to predict exactly what the effects of a truncation will be when the web service must still produce a valid SDMX-ML instance.

It is the responsibility of the querying service to adjust the query and re-send it to produce a non-truncated response, if that is what is needed.