Interim Report: DRAFT Schema Development Project

National Weather Service, Office of Hydrologic Development & Apex Digital Systems, Inc.
May 13, 2005

Introduction

The Hydrology XML Consortium (HydroXC) is an open, independent group of organizations working for the advancement of hydrologic forecasting through open communication and data sharing. The Office of Hydrologic Development (OHD), part of NOAA's National Weather Service, has provided initial funding for HydroXC activities and defined an initial project for the Consortium. With this first project, the Consortium leadership intends to create an initial version of a hydrology-specific markup language based on XML, with which participants in the hydrologic forecasting community can begin to exchange data more effectively.

This document contains an update of the project to develop a draft HydroXC schema, an overview of the schema as it has been developed so far, as well as an overview of the next steps the Consortium will take toward initial implementation of the schema.

Background

On September 29th, 2004 the initial participants in the Consortium met at the NWS/OHD offices in Silver Spring, with a good number joining via conference call. Several team members of Apex Digital Systems, Inc., the software firm charged with the technical aspects of developing the XML schema, helped to facilitate the meeting.

The group affirmed the need for standardized structures for exchanging hydrologic data with each other, and discussed current experience with data modeling with XML. As a next step, the participants agreed to provide related information to Apex as preparation for the work on creating an initial draft schema. The participants further recommended using the NWS' Standard Hydrologic Exchange Format (SHEF) as a baseline for the new schema, as SHEF is in broad use in the hydrologic community today.

As a result, several members of the group provided Apex with samples of work related to XML and/or the classification of hydrologic data. Since the group had indicated its wish to build from the existing standard SHEF, Apex has developed an initial schema by using SHEF and the materials provided as a starting point. The schema fully supports SHEF and several of the other structures provided by some of the consortium members. Very importantly, the schema also supports a variety of more complex data structures than SHEF can currently support, but which are important for more recent forecasting tools.

Based on an early review of the schema, Apex and members of the Office of Hydrologic Development concluded that it will be important to gather several interested Consortium members for a hands-on workshop during which actual data will be encoded into the proposed schema to test its viability. A more complete presentation of the schema in the context of a conference will be reserved until this initial testing and evaluation is complete, and until several Consortium members have used the schema on a test basis. The proposed schema is a draft and should be viewed as such; we expect it to change through review by the Consortium.

Architectural Considerations

During the Apex team's architectural work for the schema, it became increasingly clear that the schema will need to provide a message-centric structure instead of a repository-centric structure. The distinction between these approaches is described further below.

A typical, normalized relational database reflects repository-centric design, in the sense that it models each object with all necessary attributes, properties and dependencies. As a result, the object structure is stable, ensuring that only acceptable, well-formed data is stored in the repository.

For purposes of flexible exchange of data sets between parties, such stability and rigidity are to a certain degree counter-productive. In these types of data exchanges, the exact data points may not have been defined, or may change on a regular basis, and the schema has to be able to adjust to such flexibility. Further, the schema should annotate as much as possible the content it transmits, so that any reviewer or consumer of hydrologic data delivered in a HydroXC-compliant schema can easily understand the data contained in a schema. Finally, it is clear that each organization uses some variant or individualized hydrologic schema, and any attempt to equalize or standardize rigid schemas across all the organizations involved will not be feasible.

As a result, a message-centric schema must provide fluid, self-documenting structures that can model any data structure without prior knowledge and design changes. In addition, the schema should model the conditions and parameters under which the data contained in the schema was created or derived. Structural consistency across schema instances is less important or relevant with this approach, provided that tags and structures are used in a minimally consistent manner.

The following section provides an overview of the structure proposed in the current draft schema. An example of a request for historic crest data is used to illustrate the sections of the schema.

Schema Overview

As discussed above, the schema should provide both the capability to self-document its structures, and capture a wide variety of data without structural restrictions. In addition, the message-centric approach implies that the schema be able to represent contextual information about the data it encodes, such as when the data was requested, how it was generated, etc.

The schema consists of three basic elements: 1. the Header Element - containing information about the message, such as data status and quality, the measurement system applied to all data in the set, etc., 2. the Input Element – specifying the requested data and the structures that should be used to return the data, and 3. the Output Element - containing the returned requested data according to the structures defined in the Input Element.

The schema employs a pattern of first defining the data structures and then providing the values to fill in that structure. This way data is always provided along side an explanation of its context. For example, in the Output Element, the requested data is returned in a ParameterDataElement and this ParameterDataElement consists of two parts, a ParameterDataStructure and a ParameterDataValueArray. The ParameterDataStructure describes the data structures used to convey the data, and the actual data are in the ParameterDataValueArray. This pattern is seen throughout the schema with the Data Structures providing the context for the Data Value Arrays.

Each of the elements is described in greater detail below. Within the descriptions we have included links to sections of an example message which illustrate parts of the design. This example was created with sample data provided by OHD and it reflects a request with returned data for historic crests for one location (see attachment A to view the complete example).

Header Element

The Header element identifies the message as a discrete exchange of data between two parties. The Header provides basic information for the dataset as a whole. Specifically, the Header element contains the following information:

- Dataset Status: reflects whether it is a test or actual exchange
- Dataset Type: reflects the type of data belonging to the dataset, e.g. observed, forecast etc.
- Dataset Quality: refers to the overall dataset and any factors that might impact its quality
- Measurement System: reflects the measurement system used for the dataset, e.g. Metric
- Language: reflects the language in which the data is provided
- Time Format: reflects the general formatting of all time data in the dataset
- Time Zone: reflects the time zone for the message
- Comment: reflects any comments the dataset creator may wish to include

Input Element

The InputElement contains data provided by the requestor. It contains information about the requestor and it specifies what data is desired as well as the format in which the requested data should be returned. Specifically, the InputElement contains:

- Requestor information: reflects both automatically collected information such as IP address and requestor-provided information
- Request Time: Reflects the time and date when the request was made
- ModelDataElement(s) define the structure(s) to be used to return the data being requested. The data provided in a ModelDataElement is the description of the way the data is to be returned; it is not actual data values. The ModelDataElement(s) contain two components, a ModelDataStructure and a ModelDataValueArray. The ModelDataStructure describes the structure used to define the data elements and data types for the return data, while the ModelDataValueArray lays out the specific data elements and data types to be used for returning the requested data.
    - In the example ModelDataElement, the ModelDataStructure indicates that the ModelDataValueArray will consist of Parameter Name/Datatype pairs. The ModelDataValueArray indicates the returned data should include Rank as an Integer, Stage in feet and the crest Date as a Date. This example contains only one ModelDataElement, though a single message might contain multiple Model Data Elements.
    - Use the following link to view the example Model Data Element. At the end of the example, a link will return you to this point of the document. ModelDataElement Example

- RequestDataElement(s) are the requests for data. They contain two components, the RequestDataStructure and RequestDataValueArrays. The RequestDataStructure defines the structure of the data request. The RequestDataValueArray is the description of the requested data.
    - In the example, the RequestDataElement is a request for historic crests for the location "CCNO1," for all available times. The requested data is to be returned using the structure defined in ModelDataElement number one. A more complex request might have multiple ModelDataElements and multiple data requests making it necessary to identify which structure should be used for returning the requested data.
    - Use the following link to view the example RequestDataElement. At the end of the example, a link will return you to this point of the document. RequestDataElement Example

Output Element

The OutputElement contains the actual returned data. In the example, the InputElement remains a part of the message to provide context for the data returned in the Output Element. The Output Element contains several components, as follows:

- Provider information: contains information about the individuals or services that created the dataset in response to the request
- Timing information: contains the time and date the dataset was created, as well as the time and date when the dataset was transmitted
- CoverageElement(s) contain information about the geographical distribution of the data provided in the data set, and specifically the point or area for the data. Within a CoverageElement the data structures for each location in the coverage and for each physical element in the dataset are described. The location information is described with the LocationType and LocationValueArrays. The requested data is returned in the PhysicalElements and is described by the ParameterDataValueArray. An OutputElement may contain any number of CoverageElements; a CoverageElement may contain any number of LocationElements; and a LocationElement may contain any number of PhysicalElements.
    - In the example OutputElement, there is one CoverageElement for the CCNO1 forecast point and it contains just one location and one physical element. In the ParameterDataElement, the data structure requested in the InputElement (with the ModelDataElement) is reproduced and then the data are provided in the ParameterDataValueArray using that structure. In this case, the returned data contains the Rank, Stage and crest Date. The Flood Stage is provided in the CoverageElement location information as reference data.
    - Use the following link to view the example ParameterDataElement. At the end of the example, a link will return you to this point of the document. ParameterDataElement Example

Next Steps

As a next step, the HydroXC organizers are planning a hands-on workshop during which participants will work on modeling their own data into the proposed draft schema. Each participant will submit several data sources used in their organization reflecting the range of internal complexity of available data. During one to two days on-site in Silver Spring, MD, participants will collaborate in the data modeling exercises to ensure that their data can be adequately represented in the proposed HydroXC-compliant XML. In addition, participants will collaborate to modify the proposed schema as necessary to ensure that it can capture the data to be transmitted.

At the end of the workshop, any changes to the draft schema resulting from the meeting will be reported to other HydroXC members. The National Weather Service will proceed to implement several pilot uses of the schema in software currently under development, possibly in collaboration with other HydroXC participants. This type of implementation is intended to provide actual ground-truth information about the value and functionality of the initial schema, and will provide concrete information for improving the schema during development of the next version.

Planning has also begun for Phase II of HydroXC's work. The goal is for members of the working group to implement the schema in trial applications. HydroXC will then look for an appropriate conference at which to share the findings of those case studies.

If you have any questions or comments about this document, HydroXC in general, and the project to develop an initial schema, please contact either of the following individuals:

Edwin Welles, NOAA, Hydrology Laboratory – (301) 713-0640 ext.121

Alia Williams, Apex Digital Systems – (301) 588-9767 ext. 100

Attachment A

Draft Schema Containing Sample Data

As a practical example, the Apex team incorporated data provided by NWS/OHD into the draft schema. Working with samples of actual data was an instructive step in the development of the schema. This exercise affirmed development up to that point and helped identify areas to extend the schema. The upcoming Consortium workshop (see Next Steps section above) will follow a similar process.

```
<!-- HEADER ELEMENT -->

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Dataset SYSTEM "\\filesp\activeclients\NOAA\NWS\XMLSchema\ApexDocumentation\DataSetFiles\UHD_CCNO1_history.dtd">
<Dataset DatasetID="CCNO1" Schema="http://noaa.gov/uhd.xml">
    <DatasetType Code="H" Name="HISTORIC"/>
    <MeasurementSystem Code="E" Name="ENGLISH"/>
    <Language Code="E" LanguageName="ENGLISH"/>
    <TimeFormat Code="MM-DD-YYYY" Description="Standard Time Format"/>
    <TimeZone Code="ES" Name="Eastern Standard Time (USA Canada)"/>
    <Comment>Comments for the entire Dataset</Comment>

<!-- END OF HEADER ELEMENT -->

<!-- INPUT ELEMENT -->
    <Input>
        <Requestor LocationCode="SSMC2" LocationName="Silver Spring MetroCenter 2"/>
        <RequestTime Date="01-25-2005" Time="15:23"/>
        <ModelDataElements Count="1">

<!-- MODEL DATA ELEMENT EXAMPLE -->

            <ModelDataElement Number="1" Name="River Stage" Schema="Stage">
                <ModelDataStructure ID="424" Name="Stage">
                    <StructureElement Item="Val1">
                        <Name>ParmeterName</Name>
                        <DataType>String</DataType>
                    </StructureElement>
                    <StructureElement Item="Val2">
                        <Name>ParmeterDataType</Name>
                        <DataType>String</DataType>
                    </StructureElement>
                </ModelDataStructure>
                <ModelDataValueArray Count="3">
                    <ModelDataValueElement Number="1">
                        <Value Item="Val1">Rank </Value>
                        <Value Item="Val2">INTEGER</Value>
                    </ModelDataValueElement>
                    <ModelDataValueElement Number="2">
                    <Value Item="Val1">Stage</Value>
                    <Value Item="Val2">FEET</Value>
                </ModelDataValueElement>
                <ModelDataValueElement Number="3">
                    <Value Item="Val1">Date</Value>
                    <Value Item="Val2">DATE</Value>
                </ModelDataValueElement>
            </ModelDataValueArray>
        </ModelDataElement>

<!-- END OF MODEL DATA ELEMENT EXAMPLE -->

<!-- BACK TO TEXT -->

<!-- REQUEST DATA ELEMENT EXAMPLE -->

        <RequestDataElement Number="1">
            <RequestDataStructure ID="424" Name="SAMPLE1">
                <StructureElement Item="Val1">
                    <Name>RequestID</Name>
                    <DataType>String</DataType>
                </StructureElement>
                <StructureElement Item="Val2">
                    <Name>RequestName</Name>
                    <DataType>String</DataType>
                </StructureElement>
                <StructureElement Item="Val3">
                    <Name>LocationID</Name>
                    <DataType>String</DataType>
                </StructureElement>
                <StructureElement Item="Val4">
                    <Name>ModelDataElement Number</Name>
                    <DataType>Integer</DataType>
                </StructureElement>
                <StructureElement Item="Val5">
                    <Name>Period</Name>
                    <DataType>String</DataType>
                </StructureElement>
            </RequestDataStructure>
            <RequestDatavalueArray Count="1">
                <RequestDataValueElement Number="1">
                    <Value Item="Val1">01 </Value>
                    <Value Item="Val2">Historic Crests</Value>
                    <Value Item="Val3">CCNO1</Value>
                    <Value Item="Val4">1</Value>
                    <Value Item="Val5">ALL TIMES</Value>
                </RequestDataValueElement>
            </RequestDatavalueArray>
        </RequestDataElement>

<!--END OF REQUEST DATA ELEMENT EXAMPLE -->
<!-- BACK TO TEXT -->
```

```
        </RequestDataElements>
    </Input>

<!-- END OF INPUT ELEMENT -->
<!-- OUTPUT ELEMENT -->

    <Output>
        <Creator LocationCode="SSMC2" LocationName="Silver Spring MetroCenter 2"/>
        <TimeStampCreated Date="01-26-2005" Time="10:20"/>
        <TimeStampTransmitted Date="01-26-2005" Time="10:25"/>
        <CoverageElements Count="1">
            <CoverageElement Number="1" ID="ZZ5506" Name="Sample">
                <LocationElements count="1">
                    <Location Number="1" TypeCode="PT" Typename="POINT">
                        <LocationType ID="2" Code="P" Name="POINT">
                            <StructureElement Item="Val1">
                                <Name>Location ID</Name>
                                <DataType>String</DataType>
                            </StructureElement>
                            <StructureElement Item="Val2">
                                <Name>Location Name</Name>
                                <DataType>String</DataType>
                            </StructureElement>
                            <StructureElement Item="Val3">
                                <Name>Location Flood Stage</Name>
                                <DataType>Number</DataType>
                            </StructureElement>
                            <StructureElement Item="Val4">
                                <Name>Comment</Name>
                                <DataType>String</DataType>
                            </StructureElement>
                        </LocationType>
                        <LocationValueArray Count="1">
                            <LocationValueArrayElement Number="1">
                                <Value Item="Val1">CCNO1</Value>
                                <Value Item="Val2">Ohio River at Cincinnati</Value>
                                <Value Item="Val3">52.0</Value>
                                <Value Item="Val4">This is test data sample.</Value>
                            </LocationValueArrayElement>
                        </LocationValueArray>
                        <PhysicalElements Count="1">
                            <PhysicalElement Number="1" Code="RS" Name="River Stage">
                                <ParameterDataElements Count="1">

<!-- PARAMETER DATA ELEMENT EXAMPLE -->

                                    <ParameterDataElement Number="1" RequestID="01">
                                        <ParameterDataStructure ID="crests" Name="Hstoric Crests" ModelDataElementNumber="1">
                                            <StructureElement Item="Val1">
                                                <Name>Rank</Name>
                                                <DataType>INTEGER</DataType>
                                            </StructureElement>
                                            <StructureElement Item="Val2">
            <Name>Stage</Name>
            <DataType>FEET</DataType>
        </StructureElement>
        <StructureElement Item="Val3">
            <Name>Date</Name>
            <DataType>DATE</DataType>
        </StructureElement>
    </ParameterDataStructure>
    <ParameterDataValueArray Count="10">
        <ParameterValueArrayElement Number="1">
            <Value Item="Val1">1</Value>
            <Value Item="Val2">80.00</Value>
            <Value Item="Val3">01-26-1937</Value>
        </ParameterValueArrayElement>
        <ParameterValueArrayElement Number="2">
            <Value Item="Val1">2</Value>
            <Value Item="Val2">71.10</Value>
            <Value Item="Val3">02-14-1884</Value>
        </ParameterValueArrayElement>
        <ParameterValueArrayElement Number="3">
            <Value Item="Val1">3</Value>
            <Value Item="Val2">69.90</Value>
            <Value Item="Val3">04-01-1913</Value>
        </ParameterValueArrayElement>
        <ParameterValueArrayElement Number="4">
            <Value Item="Val1">4</Value>
            <Value Item="Val2">69.20</Value>
            <Value Item="Val3">03-07-1945</Value>
        </ParameterValueArrayElement>
        <ParameterValueArrayElement Number="5">
            <Value Item="Val1">5</Value>
            <Value Item="Val2">66.30</Value>
            <Value Item="Val3">02-15-1883-</Value>
        </ParameterValueArrayElement>
        <ParameterValueArrayElement Number="6">
            <Value Item="Val1">6</Value>
            <Value Item="Val2">66.20</Value>
            <Value Item="Val3">03-11-1964</Value>
        </ParameterValueArrayElement>
        <ParameterValueArrayElement Number="7">
            <Value Item="Val1">7</Value>
            <Value Item="Val2">65.20</Value>
            <Value Item="Val3">01-21-1907</Value>
        </ParameterValueArrayElement>
        <ParameterValueArrayElement Number="8">
            <Value Item="Val1">8</Value>
            <Value Item="Val2">64.80</Value>
            <Value Item="Val3">04-18-1948</Value>
        </ParameterValueArrayElement>
        <ParameterValueArrayElement Number="9">
            <Value Item="Val1">9</Value>
            <Value Item="Val2">64.70</Value>
            <Value Item="Val3">03-05-1997</Value>
```

```
                                            </ParameterValueArrayElement>
                                            <ParameterValueArrayElement Number="10">
                                                <Value Item="Val1">10</Value>
                                                <Value Item="Val2">63.60</Value>
                                                <Value Item="Val3">03-21-1933</Value>
                                            </ParameterValueArrayElement>
                                        </ParameterDataValueArray>
                                    </ParameterDataElement>

<!-- END OF PARAMETER DATA ELEMENT EXAMPLE -->
<!-- BACK TO TEXT -->


                               </ParameterDataElements>
                           </PhysicalElement>
                        </PhysicalElements>
                    </Location>
                </LocationElements>
            </CoverageElement>
        </CoverageElements>
    </Output>

<!-- END OF OUTPUT ELEMENT -->

</Dataset>
```