

A Proposal for Aggregating DAIS Service Requests

Corresponding author: Greg Riccardi, riccardi@cs.fsu.edu

1	Conceptual Architecture for Request Aggregation.....	1
2	Datasets	2
3	Representing Service Requests and Responses as Dataset Elements	3
4	Details Remaining to be Specified.....	4

This document proposes a strategy for grouping multiple requests for a single service into one request and coordinating the response to the collection of requests. The heart of the proposal is the introduction of the *dataset* document that provides a syntax for grouping requests and responses. Both requests and responses are transmitted between client and server as dataset documents.

When a client needs to send multiple requests to a single service, those requests can be collected and delivered to the service as a single request with no major increase in the complexity of the client or the service. Both client and service benefit from a reduction in the number of roundtrip request-response activities. The service, in addition, may see a significant reduction in authentication and authorization processing. The client benefits from the increase in the structural and semantic regularity of the response document. The service may reap additional benefits in its ability to optimize its processing because of the availability of multiple requests.

Once adopted, the request collection strategy will provide a foundation for interactions between a variety of complex service clients including dataflow engines, computational services, and data services. The extensible dataset schema will allow services to offer variations and extensions to the basic model.

1 Conceptual Architecture for Request Aggregation

Request aggregation is based on a document model of service interaction. That is, an aggregate service supports a porttype with a single method. The method accepts an XML document as its argument and returns an XML document as its result.

An aggregation of service requests is represented as an XML document that is the argument of a call on the method of the service. In this document, the method is *perform* and the XML document is a *dataset*. The response to the request is also a dataset document.

A dataset is an XML document that is comprised of a sequence of named elements, each with its own schema. A service request dataset contains one or more *request elements*. Each request element represents a method call on one of the methods of a service. The request element includes the method name and its parameters. Similarly, a response element represents a response to a request element. It contains the results of the method call.

Request aggregation allows a collection of service requests to be sent to a service as a single unit. In its simplest version, the requests form a sequence and are executed in order. The service

decomposes the request dataset and executes the requests one at a time. A response dataset is constructed from the responses of each request. If a request fails to execute properly, the corresponding response element will contain the appropriate error messages. Failure of any request terminates the execution of the sequence, but should return the results of all completed requests.

Extensions to request aggregation might support the execution of the requests in parallel or as a single transaction. Extensions might also specify different strategies for error processing. A later section demonstrates how datasets can contain data elements that can be referenced by request or response elements.

2 Datasets

The dataset is a packaging mechanism that allows various items of data, required as inputs and outputs to data intensive activities, to be collected and handled as a unit. Data collections are common in data access and integration, but also occur in computational and data management applications. Therefore the definition of datasets is likely to be of interest beyond the scope of the DAIS working group.

It is helpful to think of the dataset as a padded envelope of arbitrary capacity into which an arbitrary sequence of separate data items of any form may be put during the creation of the dataset. That dataset is then sealed, and may be stored, moved or copied. Then another process may access the dataset and obtain the data items that were placed in it. A further discipline on datasets is that their contents are described, so that a recipient can inspect a dataset to discover what it contains and how to access the information.

An additional property used by data access and integration services is that the dataset construct is recursive, that is, a data item in a dataset may itself be a dataset.

The motivations for datasets in this form are:

- To provide a convenient mechanism for sending and storing heterogeneous collections of values, so that developers of systems and applications are encouraged to group together values to be managed, stored or moved.
- This in turn, will reduce accumulated latency, handling and management overheads.
- To meet the requirement to deliver the same information as is transmitted for a variety and extensible set of possible values and mechanisms for describing values – the latter is helpful in permitting established practices to continue.
- To separate the issues of data movement, storage, etc. from the issues of constructing or accessing a message.

Libraries or objects for efficiently creating and accessing datasets and efficient mechanisms for moving datasets will likely be included in implementations of datasets.

A dataset consists of a *header*, a *summary*, a *body* and a *tail*, as illustrated in Figure 1. The header will indicate that it is a dataset, which version of dataset formats it complies with, and a hard-to-forge indication of where it originated. The summary will normally say how many data items are in the body and give their identifiers. The exception may be datasets which are effectively endless streams of data items. The body holds the sequence of data items as described below. The tail

holds sufficient information to indicate that the dataset is complete and has a low probability of having been corrupted.

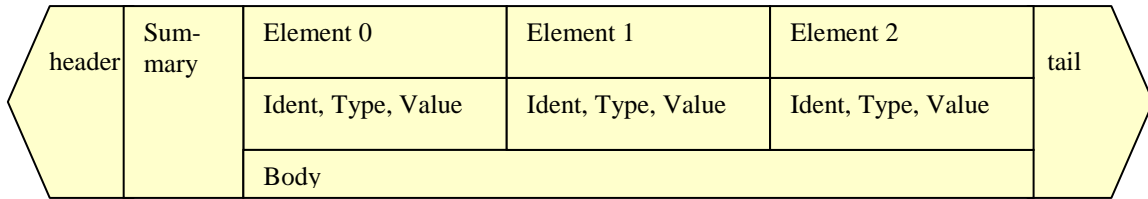


Figure 1 – the internal organization of a dataset

A *dataset element* is a self-describing value. It has an *identifier*, unique within this dataset, a *type* and a *value*. The value is often composite, containing for example a schema that describes the specific value, an indication of how the value is encoded and then the value. Any of these which can be inferred from the type may be omitted. This permits separate and incremental definition of types and their formats by groups working contemporaneously.

An alternative organization for dataset element types is to include the full schema for all of the dataset elements as part of the summary section of the dataset document.

3 Representing Service Requests and Responses as Dataset Elements

Particular conventions prevail for the datasets used in DAIS as inputs and outputs in portType operations, e.g. a request message must contain at least one data request as its first element, its value must match the data model and activity requested, and so on. Similarly a response must contain a status response corresponding with each data request in the request message.

The request dataset, as illustrated in Figure 2, must begin with a data request element recognizable by being one of a set of specified types. Figure 2 shows a dataset with 2 request elements. Each request is a call to the SQLQuery method of the service. Each element is an independent service request. Figure 3 shows a response dataset that includes a response dataset element for each request element of Figure 2. The type attribute of each element is shown as a schema name, but would more properly be a URL for the schema or the schema itself.

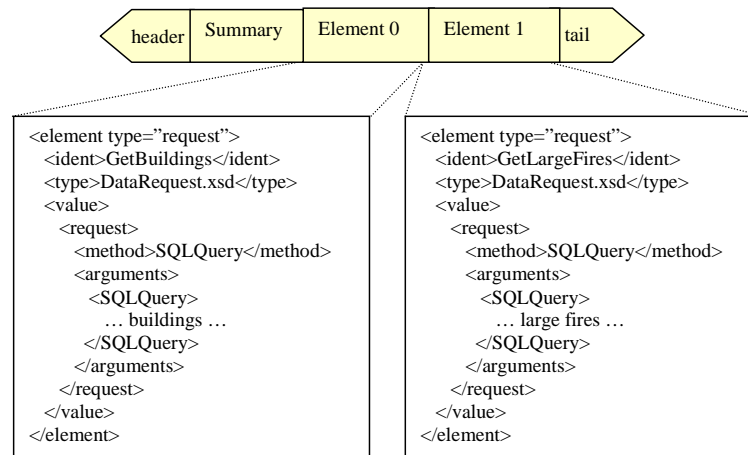


Figure 2 – An example request dataset

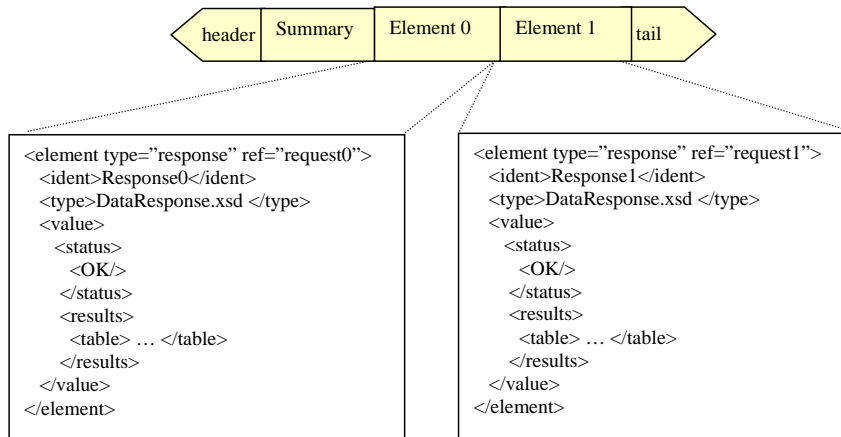


Figure 3 – An example of a corresponding response dataset

In the response dataset the first data item is the response to the first request and the second item is the response to the second request.

4 Details Remaining to be Specified

This proposal does not include the schema that defines dataset documents. Many decisions must be made about the content of schemas, about naming of dataset elements, etc.

The proposal also fails to be specific about how the arguments of service calls should be represented. This is rather a general problem of representing service requests as XML documents and not a problem that is specific to request aggregation. It is likely that the SOAP schema provides an adequate representation for treating service requests as XML documents.

Another document, in preparation, will present the way that update data can be included as dataset elements in request documents. That document will propose a mechanism for allowing a request element to refer to data contained in other elements and how response data can be represented as dataset elements. This document will extend these ideas to suggest ways of specifying simple data transmission protocols within dataset elements.