MACDDAP – Translation Services Design

Version: 1.2

Date: 16th April 2009

Author: Jason Lohrey

Title: Chief Technology Officer, Arcitecta

Contact: jason.lohrey@arcitecta.com

Revision History

Version	*Date*	*Author*	*Description*
0.1	16-Mar-2009	Jason Lohrey	Initial
1.0	30-Mar-2009	Jason Lohrey	Completed the Graphical User Interface section. Added sections 5 (authentication and authorization) and 6 (reporting).
1.1	15-Apr-2009	Ray Williams, Jason Lohrey, Pauline Mak	Incorporated minor spelling and grammatical corrections after review by Ray Williams. Emphasize that the translation is for metadata, not data.
1.2	16-Apr-2009	Jason Lohrey	Input from Pauline Mak - Expanded the functional translation example to include geospatial datum transform. Added a paragraph defining the value of metadata transformation without corresponding data transform.

Table of Contents

1. Introduction

There are hundreds, probably thousands, of millions of sets of data acquired by a raft of research disciplines over many years that need, or will need to be, conformed to different standards for cataloguing or describing those data sets. That process is ongoing, as new standards continue to emerge into the future.

When the target form has the same, or a subset of information, then this translation can generally be automated. However, the output form is often a superset of the input form. This case is increasingly more prevalent as the value of contextual metadata is recognized. Consequently, in order to produce the output form, the data (which may have been acquired decades ago, or by devices and formats that do not have the requisite or desired context) will require the provision of additional information.

Typically, the supply of additional information requires human intervention. Whilst it may take only tens of seconds to a few minutes to enter the required information, this cost becomes prohibitive when multiplied by hundreds of millions of data sets. It can be prohibitive, and perhaps error prone, with even a few thousand translations.

Since the inputs and outputs can be arbitrary, this may at first seem an unbounded problem. However, the act of translation is a process that has common patterns, invariant of the types of inputs and outputs. There are opportunities to optimize and consequently minimize the amount of manual, and costly, human intervention.

The goals of the Translation Service are to minimize the effort and cost required to translate data from one form to another.

Translation Services are a component of MACDDAP project and has been funded through the Australian Government NCRIS programme and sub-programme NeAT.

1.1. What are the "Translation Services"?

Translation Services provide a community accessible Internet accessible suite of self-service tools to facilitate the translation of data sets and associated metadata to some other "form".

Example forms might include:

- Open Geospatial Consortium (OGC) formats
- ISO 19115 metadata standards
- NetCDF datasets
- Etc.

Translation Services are constrained to descriptive and contextual information (metadata), rather than the non-descriptive data such as measurements, acquisitions, samples, etc[1].

Translation Services are accessible via browser based and command-line interfaces to:

- Allow the creation of one or more *profiles* for translations between different input and output forms. A profile:
  - Is associated with a user or organization.
  - Allows the selection of:
    - Input and output types, from an extensible set of supported formats.
    - Specification of data sources and destinations from an extensible set of supported repository types.
  - Specifies additional metadata and vocabulary mappings required to map to the output format.
- Translate an input data source using a specified profile.
- Register data sources for automatic harvesting and translation.
- Register new handlers for input and output formats.

As indicated, the types of inputs and outputs data could be arbitrary. The Translation Services will provide an open Application Programming Interface (API) to allow the development and plugging in of handlers and transformers for different types of input and output data. The Translation Services project will provide some exemplar transformations using the API with the expectation that additional handlers will be developed over time as required by stakeholders.

1.2. Technologies

The Translation Services use the following technologies:

- Browser-based interfaces utilize JavaScript
- The Translation Service server is written in Java. Plug-in components utilize Java. It is possible to use the Java Native Interface (JNI) to make direct library calls to any C/C++ library. It is also possible to execute any other application by forking a sub-process.

[1] Whilst it is out of scope for this project, it is worth considering whether it is feasible or not in the future to extend the techniques applied to metadata to all data within a dataset as that will test the generality of the translation framework.

2. Translation Framework

Translation Services provide a framework that encapsulates the common aspects of the process of translation of metadata. It enables people to establish and reuse contextual information across multiple translations.

An individual often deals with data generated:

- From similar sources,
- From the same authors,
- In similar formats.

The information that is common should be captured and saved for reuse with other sets of data. That is, there will be some initial effort to populate missing data for a translation that if saved, can be re-used for future translations.

2.1. Translation Components

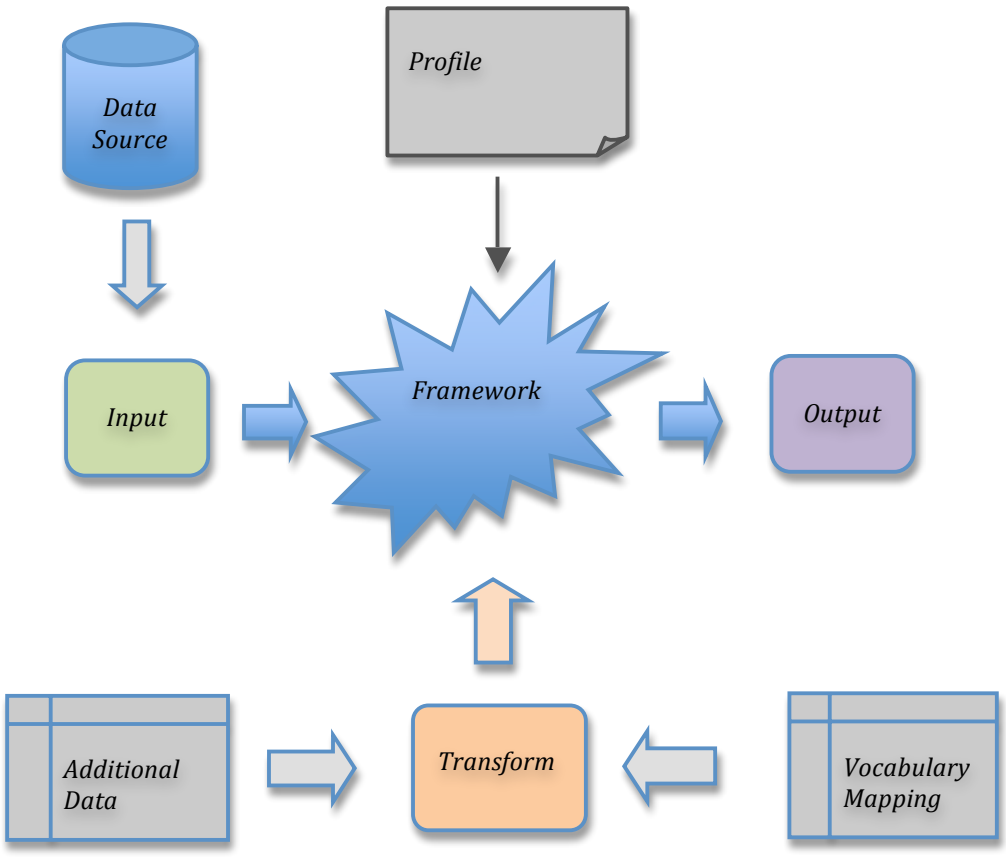

Figure 1 – Translation Components

Whilst the process of translating might be obvious, it's worth identifying the components shown in Figure 1 – Translation Components and considering the dependencies therein. Since there are an arbitrary number of inputs, outputs,

and transformations, the Translation Framework should not "know" about these formats. Rather, an *Input* or *Output* handler that is specific to that format should provide the capability for reading and writing a data format. Equally, the input and output handlers should be independent of each other.

The process of generating an output is handled by a *Transform*. A transform is a set of instructions that:

- Defines the set of data from the *Input*,
- Defines what *Additional Data* should be included,
- Defines any changes in the *Vocabulary Mapping*, and
- Specifies the recipe for combining the input and additional data to produce the output.

In order to select the correct additional data and vocabulary for a transformation, a profile is determined by either:

- Manual selection by the translation requestor, or
- By association with the source of the input.

2.2. Component Interchange Format

Metadata must be represented in some common, intermediate, format that can be passed from an *Input*, *Additional Data* and *Vocabulary* to the *Translation Services Framework*, *Transform* and *Output*. The initial inclination might be to suggest that the Extensible Markup Language (XML) is used for interchange. However, XML has some disadvantages, namely:

- It is a text-based standard and will be more verbose and less efficient than binary representations. This can be a problem where there is a lot of metadata.
- XML requires each element to be closed with a matching end-element which makes it harder to stream very large documents. That is, XML documents are typically fully constructed before passing to the receiver[2].

Rather than XML, the Translation Services will use a hierarchical key/value pair stream (*DataStream*) to represent the Metadata for interchange between the components. The interface to this data representation for does not prescribe a particular storage format – it is possible to implement the interface using XML, or some other binary representation. Like XML, the DataStream supports:

- Key/Value pairs
- Hierarchical data representation

The DataStream representation is data type independent. Whilst the underlying representation may be typed, every type must be convertible to/from a string. Transforms will not be concerned with type, but rather mapping of structure and vocabulary.

[2] There are some programming interfaces that allow streamed XML, but they are not the norm.

If required, a DataStream may be easily converted from/to XML. This also allows a given transform to utilize the Extensible Stylesheet Language (XSL) Transformations (XSLT).

2.3. Transforming Data

A transformation, T, generates some Output, as a function of the pair:

Output = T(Profile,Input)

The *Profile* defines:

Profile = (Input type, Transform, Additional Data, Vocabulary Mapping, Output type)

An *Input* will produce a *DataStream* that is passed to a specified *Transform*. The transform will apply some form of mapping to create the set of elements required by an *Output*. The *Transform* passes a *DataStream* to the *Output*.

It is possible that no transformation of the *DataStream* is required in order to generate an *Output*, in which case the *Transform* simply passes the *DataStream* through unmodified.

The *DataStream* may or may not have sufficient elements to generate the specified *Output*. In this case, additional data will be required. Provided the *Profile* has one, the *Transform* will be supplied with a *DataStream* from *Additional Data*. Where no additional data exists, or is incomplete, the transformation will not proceed and the requestor will be notified of the missing elements. The requestor may then augment the *Profile* or create a new *Profile*.

The names of elements and their values may need to be transformed for an *Output*. For example, one Input may describe a geographic coordinate using "Lat" and "Lon", whereas the Output may require "Latitude" and "Longitude". In this case, the name of the element will need to be changed. That mapping is contained in a *Vocabulary Mapping*.

Similarly, the metadata values may need to be converted. There are two kinds of conversions to consider:

(a) A simple term mapping – for example, an *Input* may use the terms "M", and "F" to represent male and female respectively, whereas a the *Output* may expect "Male" and "Female".
(b) A functional mapping – for example, changing upper and lower bounds for temperature from degrees Celsius to degrees Kelvin, or changing the datum for a spatial extent from NAD 27 to WGS84.

(A) is really a special case of (b).

The (a) case can be handled with a simple dictionary. The (b) case will require the ability to provide an arbitrary set of instructions for transformation. The Translation Services expose an API to allow Transform creators to plug in software to perform functional mappings.

It should be possible for functional mappings to be individually registered, enabling a transform creator to reuse "stock" transforms; e.g. conversion from degrees Celsius to degrees Kelvin.

The translation of values for the metadata does not transform the underlying data (but could be considered as future enhancement). However, this does allow for the important capability for discovery of resources using a common terminology. The conversion of the data to another form is an activity outside of the scope of this project – which may not be required since applications dealing with the data may already perform those conversions.

3. Profiles

The power of the Translation Services lies in the ability to build up and save contextual information for use across multiple data sets and for future reuse. That contextual information is stored in a *Profile* that is associated with a registered "user" within the system[3].

Profiles can be later used/reused for sets of data with the same characteristics. For example, profiles can be used for data sets sharing the same authors, same custodians, fixed vocabulary mapping etc.

Since a sub-set of those characteristics might be common to different sets of data, Profiles may be created from other Profiles to aggregate settings. For example, there might be a profile that specifies the author and custodian, and a sub-profile the deals with the specific type of data set.

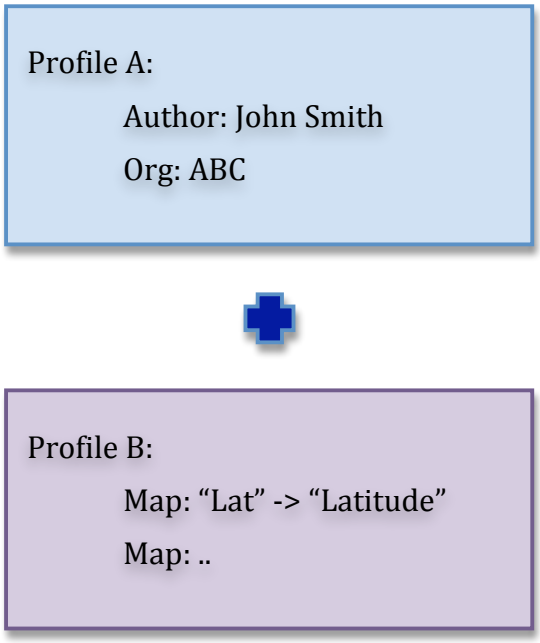

Figure 2 - Aggregating Profiles

A given Profile may aggregate multiple other Profiles.

The ability to aggregate allows the specification and supply of additional metadata and vocabulary mappings to be modularized.

The Profile is an association of the following:

Input Type	The type of input handled by this profile. For example, the input could be "NetCDF", "NITF", etc. There must exist an input handler for the type. This is plug-in code supplied to handle that format.

[3] A "user" could be a person or another system.

Transform	Specifies the relationship between the inputs and outputs. The transform may specify the inclusion of *Additional Data* and may utilize a *Vocabulary Mapping* to change the names (and values) of elements within the Input.
Additional Data	Is a set of metadata elements that must exist in the *Output*, but need not exist in the *Input*. The *Additional Data* may override existing elements in the *Input*. There may be no extra elements in the *Additional Data*.
Vocabulary Mapping	Defines changes in element names and mappings for element values. The mapping may be achieved using a lookup table (LUT), or by some software that computes the mapping.
Output Type	Specifies the type of output to be generated. There must be an output handler for the type. Output handlers are plug-in modules.

4. Interfaces

Two methods for accessing the services will be provided:

- Command-Line Interface – provides the ability to create profiles, initiate transfers from the command-line.
- Graphical User Interface – a browser-based interface. This is expected to be the primary method of utilizing the services.

These interfaces are described in more detail as follows.

4.1. Command Line Interfaces

The command line interface requires an executable (client) that provides the same capabilities as the Graphical User Interface, but is simply accessible from the command line without a graphical interface.

The command line interface allows translation processes to be incorporated into other "back of house" workflows without human intervention. As with Graphical User Interfaces, the command-line interfaces require the client to "log into" the Translation Services server in order to establish account settings.

For more information on the functional services and capabilities, refer to the section 4.2 Graphical User Interfaces.

4.2. Graphical User Interfaces

A Graphical User Interface exposes *Profiles*, *Transforms* etc. through a web-browser interface.

A typical interaction with the system is as follows:

- Person logs into the system – if they don't have an account then they will then they will need to create one. An "account" is required to maintain personal settings (this is a free service). Account usage is also a good way of determining whether people are or are not using the service.
- Create one or more *Profiles* that specify the transformation from one input to another. This process occurs as follows:
    - Select an existing Profile as a template, or create a new one.
    - Specify the Input Type – this is selected from a pull-down menu of the registered type handlers in the system. That list will be extended over time.
    - Specify the Output Type – this is selected from a pull-down menu of the registered type handlers in the system.
    - Specify any Additional Data to be added. The additional data may be hierarchically organized.
- Specify a source data set, and associate it with a Profile. A source data set may be provided by one of the following means:
    - File upload
    - URL reference (URL must be in a form that can be handled by the *Input Type*).

- Specify the location of the output, which may be generated by one of the following means:
    - File download
    - Requesting the *Output Type* handler to consume the output. That is, an *Output Type* handler may have the capacity to write the output back to some other location (e.g. database, data server, etc.).
- Optionally save the combination of Source, Profile Settings and Output (as another Profile). This profile may be utilized the next time the user visits the system.

As indicated earlier, a profile may define a subset of the required transformation. A profile can be composed of other profiles, with specific overrides and supply of information that may not have been in the used profiles. That is, a profile may be incomplete, but contains common information.

Graphical user interfaces will also enable administrative control of the system (see section 5.3 Administering Translation Services) and for generating reports (see section 6 Reporting).

4.3. Application Programming Interfaces

Application Programming Interfaces (API) are required to enable the development and installation of additional:

- Input Type handlers
- Output Type handlers
- Functional transformations

The Translation Services Framework will be written in Java. It will expose interfaces to:

- Add/List/Remove Java Archives (JAR) containing *Input Type* handlers
- Add/List/Remove Java Archives (JAR) containing *Output Type* handlers
- Add/List/Remove Java Archives (JAR) containing functional transformations.

A single JAR may contain more than one type of plugin capability. All plugin code will require descriptive information:

- Version
- Author
- Formats handled
- Usage notes
- Caveats, if any.

That is, there must be enough information for a Translation Service user to fully understand whether or not to utilize the handler.

Even though the application development platform is Java, it is possible for Java to bridge to any other technology.

5. Authentication and Authorization

5.1. Translation Services

The translation services are a publically and freely available. Therefore, a potential user will be able to setup an account in the system without reference to any system administrator. Self-served accounts will be able to make use of the Translation Services, except the installation of additional plugin software (see below).

5.2. Extending the Platform

The ability to add new code to the platform must be a "controlled" process requiring authority to do so. If not, the platform may be overloaded with rogue software that renders the service inoperable.

In order to extend the server through additional format handlers, application developers must be granted an *application developer* role.

The Translation Services Administrator must approve the addition of new software before it is available to translation services users. Such approval may be granted on a case-by-case basis, or the developer may be pre-approved for all developed software, without the ongoing need for administration.

5.3. Administering Translation Services

The server will require administration to control who and how people can add new services to the platform. A specific role of *system administrator* has the authority to grant the *application developer* role to users of the system.

Administrators can also create other administrators.

The contact details for administrators will be clearly visible/available from the Translation Services interfaces to enable would-be developers to request access to the system.

6. Reporting

For the purpose of utilization, and capacity planning, Translation Services will be able to report on the following:

- The number and details of registered users (and their level of authority within the system)
- The number of translations performed.
- The types of translations performed.

Reports will be run ad-hoc. Reporting should be available to any user of the system. However, only administrators will be able to see the authority level of other users.