The TMS Interoperability Protocol Package (TIPP)

V1.4.1, March 30, 2012

Document History

Revision	TIPP Version	Date	Author	Changes
2	1.0.1	14/11/10	Sven Andra	D: non transitive package D: Versioning convention D: IDs
3	1.2	26/6/11	Chase Tingley	Align with schema updates
4	1.2	28/6/11	Chase Tingley	Updated naming (TIPP), added Relationship to Other Efforts section
5	1.3.0	19/8/11	Chase Tingley	Updated based on feedback from Yves Savourel, Christian Lieske, and David Filip.
6	1.3.1	22/8/11	Chase Tingley	Updated date format and path length restriction sections based on feedback from Micah.
7	1.4.0	1/3/12	Chase Tingley	Major revisions to task structure.
8	1.4.1	30/3/12	Chase Tingley	Updated based on feedback from Alan Melby and IN! members.

Introduction

The TMS Interoperability Protocol Package (TIPP) is an information container that allows the exchange of information between different Translation Management Systems (TMS).

This reference guide describes the package, the package description and standard methods to interact with the package. The representation of the content itself is not described in this document.

Why TIPP?

As the translation ecosystem matures, the supply chains involved in the process of translation are becoming increasingly complex. During the translation lifecycle of a single piece of content, it may be touched by multiple tools controlled by several different parties. Where once TMS usage was a rarity, it is now common to encounter multiple TMS systems on the journey between source and target locale.

There currently exist two major ways to pass content between these different systems:

- Proprietary solutions specific to a particular TMS or toolset from a single vendor, supporting content as well as additional metadata.
- Standalone file formats such as XLIFF or TMX that can represent a subset of the necessary data, but not the associated metadata or supplemental information required in a translation process.

Neither of these solutions is sufficient for general-purpose use. Proprietary solutions tend to be poorly-documented and are not general-purpose enough to operate in the heterogenous environments that characterize the modern translation industry. The existing standards, while an important component to an interoperability solution, are only solving parts of the overall problem.

What is TIPP?

TIPP is a transport layer for translation data and metadata.

TIPP is designed to replace the ad-hoc system of FTP sites, zip files, and impromptu packaging formats that inhabit the exchange points between different nodes in a translation supply chain. TIPP simplifies these exchanges by providing a common packaging system for translation data and metadata. It can be used to provide strong interoperability guarantees, while also allowing enough extensibility

to be adopted for new uses.

To do this, TIPP utilizes a couple of core concepts.

- **Task Types**. Each TIPP is associated with a single task type. The TIPP may represent either the request to complete a task, or a response to a previous request. The TIPP specification defines several built-in tasks, representing different core behaviors in the translation supply chain. However, additional tasks may be defined by third parties. Different tasks may make different interoperability guarantees.
- **Manifest**. Each TIPP contains a single metadata file called a manifest. The manifest identifies the TIPP's task, whether it is a request or a response, and certain types of generic metadata that are common across all tasks. It also enumerates the contents of the TIPP payload. It also contains security information related to the encryption and digital signing of the payload.
- **Payload**. Each TIPP contains zero or more files known as the payload. These files represent user data, tool data, etc. The number and types of files supported varies by task. The payload is stored in an internal container within the TIPP, and is optionally signed and encrypted.

How does TIPP improve interoperability?

TIPP improves interoperability in two major ways.

- **Standard representation for common metadata.** The TIPP manifest provides a standard way to encode certain pieces of metadata that are useful regardless of task, including:
    - Request/response semantics, including information about success or failure
    - Information about the package creator
    - Unique package identification
    - A standard way to discover TIPP contents without opening the entire package
- **Task types may impose their own requirements.** TIPP tasks may mandate additional restrictions on the types of files that a given task may use, as well as the processing expectations for handling a TIPP request of a given task. The highest level of interoperability can be achieved by using the

standard Translate task type, while the Generic-Bilingual and Generic-Translate task types allow for more flexible use in exchange for less guaranteed interoperability.

Is TIPP enough to guarantee interoperability on its own?

No. TIPP is just a standard way to represent certain types of data exchange. However, that is only one piece of the puzzle. There are other types of data involved in a translation supply chain that also need standardized representations, as well as well-defined rules for how the data should be interpreted and handled.

The Interoperability Now! working group has developed TIPP to work best in conjunction with other formats such as XLIFF:doc.

Goals of TIPP

TIPP has been designed to be open-ended: as a package format, it can be adopted for other uses that are not explicitly described here. However, the task types described in this document are meant to address a particular set of use cases:

- When used in conjunction with XLIFF:doc, a strong guarantee of interoperability between TMS systems.
- When used in conjunction with other bilingual formats such as XLIFF 1.2 or gettext (.po), a lesser guarantee of interoperability between TMS systems.
- In the absence of well-defined bilingual formats, the ability to function as a basic packaging mechanism between TMS systems, or between a TMS and a CMS.

Task Types, Extensibility, and Compliance

The TIPP specification defines a number of Standard Task Types:

- Translate-Strict-Bitext
- Translate-Generic-Bitext
- Translate-Native-Format
- Prepare-Specifications

However, it is expected that the TIPP package structure may be adopted for additional purposes beyond the standard task types. Custom Task Type may be defined to handle these specific use cases. Some Custom Task Types may describe internal processes that are not applicable outside of a single toolset or environment, while others may describe more general processes not covered by the Standard Task Types, such as review or automated quality assurance processes.

Defining a Custom Task Type

For each custom task type, a human-readable task type definition should be created. The task type definition should include:

- A URI that uniquely identifies the task type.
- For each supported Object Section, a unique identifying URI and a description of the Object Section, including supported file types.
- A description of the criteria for returning Success or Failure in a Response TIPP of that particular task type.
- Documentation of any related processing expectations.

While there is no required format for a custom task type defintion, it is proposed that the format used in Standard Task Types be used as a model.

Evaluating Compliance with the TIPP Specification

The extensible nature of TIPP raises obvious questions about the meaning of "TIPP-compliance" in the context of interoperability. Additionally, it is important to note that TIPP could potentially be used by systems for which the Standard Task Types are meaningless - for example. Therefore, a two-tiered system of compliance is proposed:

- **Basic TIPP Compliance**. In order to be TIPP Compliant, an implementation must be capable of reading and writing valid TIPPs, as enforced by the XML schema and conforming to the additional restrictions described in the Reference Guide.
- **TIPP Standard Task Type Compliance**. In addition to the requirements described under Basic TIPP Compliance, TIPP Standard Task Type Compliance requires than a tool be capable of processing request and response TIPPs of the types defined in Standard Task Types, and taking action appropriate to the tool's role in the supply chain.

Additional Standard Task Types

Future versions of the TIPP specification may add additional Standard Task Types.

Versioning of this Reference

An important part of the approach for the TMS Interoperability Protocol Package is an agile und fast implementation of the first version of TIPP. This reference might therefore change quickly over time and it is important to understand the compatibility between the versions.

The version of this reference (which will also be included in every package) consists of three numbers, period-separated, starting with 1.0.1.

The first number describes the major version: Differences between major versions might lead to a lack of compatibility.

The second number describes the version of the manifest format: All basic changes in the *manifest.xml* will lead to a new second digit.

Changes in the third number imply changes to the specification, but no changes to *manifest.xml.*

Glossary of Terms

Envelope	The outermost container in a TIPP. In the current implementation, this is a ZIP archive.
Manifest	An XML file that contains all metadata needed to process a TIPP.
Package Object Container	The portion of a TIPP that stores the package objects. The Package Object Container is represented as an (optionally encrypted) ZIP archive located within the Envelope.
Payload	Alternate name for Package Object Container.
Object	An individual file stored in a TIPP. Future TIPP versions may allow Objects to be referenced externally (eg, via HTTP), rather than contained directly in the Payload.
Request	A TIPP that defines a particular localization task to be completed.
Response	A TIPP that contain the results of an attempt to complete the localization task defined by a given Request.
Task Type	A set of expected behaviors and semantics for a TIPP request/response pair. The TIPP specification defines several basic task types, but others can be defined by third parties.

Task Types, Requests, and Responses

Every TIPP defines either a *request* or a *response* for a particular *task*

type. Each task type describes a single operation to be performed as part of a localization process. A request TIPP specifies the task type and the resources required to perform task. A response TIPP contains the results of an attempt to process the request, along with any resources that are produced in the course of processing the request.

For example, the "Translate" task type defines the translation of a single XLIFF:doc file. A package with the "Translate" task type will contain:

- Metadata that is common to all TIPPs, such as source and target locale. This metadata is stored in the manifest.
- An XLIFF:doc file containing content to be translated.
- Supplemental resources associated with translation, possibly including:
    - A Structured Translation Specification (STS) file
    - A TMX file with additional translation memory matches
    - Supplemental resources used for HTML preview, such as stylesheets or XSLTs
    - A style guide or other reference materials

While Interoperability Now defines a set of standard task types that must be supported in order to be fully compliant, the format is extensible and additional task types may be defined by third parties.

Each Task type definition must specify:

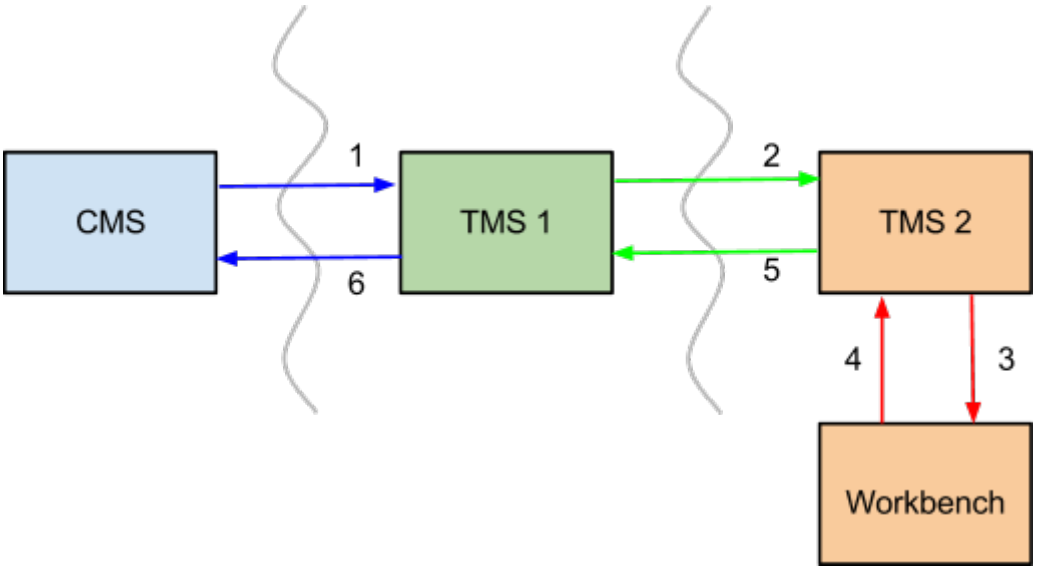
- An identifying URI for the Task type.
- The allowed Package Object Sections, each identified by a URI. A Task type may specify different Package Object Section requirements for its Task Request and Task Response.
- For each Package Object Section, the allowed "Type" values for Objects in that section. A Task type may specify different Object Type requirements for Task Request and Task Response.
- Processing expectations for the Task type, including
    - The conditions under which a Task Response may indicate "Success".
    - What modifications to the Package Objects, if any, are allowed or required in the Response.

Each task type may specify the types of package objects it supports, along with defining processing expectations for supporting implementations.

The standard task types are defined in Standard Tasks.

How TIPP Fits into a Translation Supply Chain

A series of TIPP request/response pairs can be used to implement data exchange at different points in a translation workflow. The diagram below demonstrates one such scenario.



In this diagram, there are four TIPP-aware tools controlled by three separate organizations:

- A translation buyer stores their content in the CMS.
- At the start of a translation cycle, they pass content to a multi-language vendor (MLV) who uses a TMS to manage their work.
- The MLV in turn passes the content to a single-language vendor (SLV), who also uses a TMS to manage their work.
- Within the SLV, a translator uses a translation workbench to perform the translation.

Using TIPPs, the translation process might look like this:

1. The CMS packages source files into a TIPP request using the Generic-Translate task type. The TIPP is sent to the MLV's TMS system, where the source files are processed through the MLV's translation workflow.
2. The MLV's TMS creates a TIPP request with the Translate task type, containing an XLIFF:doc file with content received from the customer. The TIPP request is sent to the SLV's TMS, where the XLIFF:doc is processed through the SLV's translation workflow.
3. The SLV's TMS creates another TIPP request with the Translate task type, again containing the XLIFF:doc representation of the original source file. The TIPP request is sent to (or downloaded by) a translator, who loads it into a workbench tool and performs the translation.
4. The translator's workbench tool generates a TIPP response

with the Translate task type, containing an updated copy of the XLIFF:doc with the translation performed. The TIPP response is sent back to the SLV's TMS, where the SLV's translation workflow is completed.

5. The SLV's TMS generates a TIPP response with the Translate task type, containing the translated XLIFF:doc. The TIPP response is sent back to the MLV's TMS, which completes its translation workflow and generates the translated target source files.
6. The MLV's TMS generates a TIPP response with the Generic-Translate task type, containing the translated target files. The TIPP response is sent back to the customer CMS.

Of course, the workflow could be considerably more complicated, involving additional stages for review, additional translation, or even automated processes such as machine translation or automatic quality checking. Additionally, if not all systems are capable of processing XLIFF:doc files, the Generic-Bilingual task type may be used for some steps.

What is important to note is that a separate request/response pair is generated for each exchange. Other than the ability of a TIPP response to identify the TIPP request to which it is responding, there is no way for TIPPs to reference each other. Consequently, TIPP does not provide a way to track end-to-end state changes across the supply chain, unless such data is tracked through a separate mechanism embedded within the TIPP payload.

TIPP Structure

A TIPP consists of two parts:

- The manifest, an XML file defined by the TIPP Schema.
- An (optionally encrypted) Package Object Container. The Package Object Container is a ZIP archive containing one or more Package Objects. The Package Object Container is also referred to as the Payload.

These two parts are contained in the package itself, which is also called the Envelope. In the current implementation, the Envelope is a ZIP archive.

There is no folder structure within the Envelope. The manifest must be named *manifest.xml.* The Package Object Container which must be named *pobjects.zip.enc* if encrypted or *pobjects.zip*, if unencrypted.

TIPP Manifest (manifest.xml)

The manifest contains information on all Package Objects included in the Container. Any Package Objects not described by the manifest can be ignored by package processors.

The manifest is an XML document conforming to the schema located

at

http://schema.interoperability-now.org/tipp/TIPPManifest-1_4.xsd

The rest of this section describes the information contained in the manifest.

Package ID

Each TIPP is assigned a globally unique identifier in the form in the form of a 128-bit Universally Unique Identifier (UUID), encoded as a URI, as described in RFC 4122.

Package Creator Information

The manifest contains information about the system that created the TIPP. This includes:

- *Organization name*, the name of the company or individual who created the TIPP.
- *Organization ID*, a URI identifying the organization that created the package. Note that this is only meant to be advisory, rather than to identify a specific system. For example, a TIPP may be created by a TMS behind a company's firewall; in this case, the URL of the company's public web site is an acceptable value for this field.
- *Creation Time*, as specified in Format of Date/Time Fields.
- *Tool Identifier*, describing the tool that created the package.

Task Information

Each TIPP is associated with a single Task Type and a single Source/ Target locale pair. Additionally, a TIPP represents either a Request or Response. Manifests for both Request and Response TIPPs include a Task descriptor, containing:

- A task type, identified by a URI.
- Source locale code
- Target locale code

TIPPs that represent a Task Response include both the original Task information and additional information specific to the Response:

- Information about the creator of the Response. This contains the same fields contained in Package Creator Information.
- A Response Message, either “Success” or “Failure”, depending on whether the task was executed successfully.
- A Response Comment, where the Responding entity may embed additional, human-readable information about the response. For example, in the case of task failure, error information may be embedded as a Response Comment.

Package Object Information

The manifest contains a complete list of all other files included in the Package Object Container.

The Package Object Container contains zero or more Sections, each of which contains zero or more Objects. Each Section is identified by

- A URI that uniquely identifies the Section.
- A short name that uniquely identifies the Section within the package, but may not be globally unique.

Individual task type definitions specify what Sections are required or allowed for a given Task.

Several pieces of information are provided about each Object:

- A Path, which provides information about how to locate the data for this Object. For File Resources, this is a path to the Object within the Package Object Container itself, relative to the containing Section. In the future, other Object types may allow for the loading of remote Resources from outside the Package.
- An Name, which may or may not be the same as the Path.
- A non-zero positive integer, called the Sequence Number. Sequence Numbers provide a way for the package creator to order the Objects within a given section in order to provide additional contextual information about the package contents.

Security

TODO: The PDF optionally contains a <Security> element with two parts:

1) Embedded XML-DSIG information for signing of the manifest itself and the payload.

2) If the payload is encrypted, information about algorithm and identifying info for any key required to decrypt it.

Package Object Container (pobjects.zip/pobjects.zip.enc)

The Package Object Container contains zero or more Package Object Sections, each of which contains zero or more Objects. The structure of the Package Object Container corresponds to the structure defined in the manifest.

If the Package Object Container is empty, it may be omitted from the TIPP.

In TIPP 1.4, the Package Object Container is represented as an embedded ZIP archive called *pobjects.zip*. If encryption is used, the encrypted ZIP archive must be named *pobjects.zip.enc*.

Package Object Sections

Package Object Sections are represented by directories within the ZIP archive. Each directory is identified according to the short name specified for that Section in the manifest.

Package Objects in a Package Object Section are represented as files within the ZIP archive, in the appropriate directory for that Section. Additional subdirectories may be created within the top-level Section directories.

Note that all Objects within the ZIP archive must follow the path and naming restrictions described in Format of Package Object Paths. As a result, implementations may need to abbreviate or normalize Object names in order to generate paths within the package. The original name for a Object should be stored in the Name field for that Object in the manifest.

Each Task definition enumerates the allowed Sections for that Task, including the identifying URI and optionally recommending a short name for use in the Package Object Container.

In any given TIPP, Sections that contain no Objects may be omitted from the Package Object Container.

Standard Task Types

This section enumerates the TIPP Tasks defined as part of the Interoperability Now! effort. Additional tasks may be defined by third parties. To be compliant with Interoperability Now!, all of the tasks enumerated here must be supported.

Translate-Strict-Bitext

The Translate-Strict-Bitext task type represents the translation of bilingual content in the form of an XLIFF:doc file. The Translate-Strict-Bitext task type is the most restrictive of the standard TIPP task types; it also provides the highest level of interoperability of any standard task type.

For more flexible translation tasks with a weaker guarantee of interoperability, use the Translate-Generic-Bitext task type.

Task ID

The Translate-Strict-Bitext task is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-strict-bitext

Supported Package Object Sections

TIPPs with the Translate-Strict-Bitext task type support the following package sections:

Section	Request TIPP	Response TIPP
Bilingual	Required	Required
Structured Translation Specification	Optional	Optional
Preview	Optional	Optional
TMX	Optional	Optional
Reference	Optional	Optional

Each of these sections is defined in more detail below.

Bilingual

The required *bilingual* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-strict-bitext/bilingual

The bilingual section must contain exactly one package object, which must be an XLIFF:doc file containing the content to be translated.

The source and target locales for the XLIFF:doc must match the source and target locales in the TIPP manifest.

In the response TIPP, the bilingual section should contain the XLIFF:doc from the request TIPP, modified with whatever changes were made during translation.

Structured Translation Specification

The optional *Structured Translation Specification (STS)* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-strict-bitext/sts

If present, this section should contain exactly one object, which should be a Structured Specification as defined by ISO 11669.

In the response TIPP, this section plays no purpose and may be omitted, even if it was present in the request.

TMX

The optional *TMX* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-strict-bitext/tmx

This optional section is used by a request TIPP to carry supplemental translation memory information to be used during the translation process. The TMX section may contain one or more files, each of which must be in the TMX 1.4 format. This data is intended to supplement the preferred matches which are carried as part of the XLIFF:doc itself. Each TMX is expected to contain TUV data for the source and target locales specified in the TIPP manifest, although it may contain target TUVs for other locales as well.

In the response TIPP, this section plays no purpose and may be omitted, even if it was present in the request.

Preview

The *Preview* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-strict-bitext/preview

This optional section is used by a request TIPP to carry supplemental files used in the preview process provided by the XLIFF:doc file. There are no restrictions on the types of files contained in this section, as the contents are determined by the implementation of the preview process.

In the response TIPP, this section plays no purpose and may be omitted, even if it was present in the request.

Reference

The *Reference* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-strict-bitext/reference

This optional section is used by a request TIPP to carry supplemental files that may be of use during the translation process. Examples of reference files include the source files, style guides, or other notes from the content creator. The Translate-Strict-Bitext task assigns no particular meaning to these files and does not expect any action of them to be performed.

In the response TIPP, this section plays no purpose and may be omitted, even if it was present in the request.

Processing Expectations

A system that generates a Translate-Strict-Bitext request package expects a basic translation process to have been performed on the XLIFF:doc that it receives back as part of a successful Translate-Strict-Bitext response. However, it is not a requirement that the translation be "complete" in order for the response to report "Success". There are several reasons for this:

- "Completion" is a subjective term when dealing with a bilingual file format, and it would be difficult to accurately evaluate it via automation. For example, in one context an empty target TUV could be considered an error, while in others it might be the intended "translation" of source text that does not appear in the translation.
- It is a common use case in enterprise environments for a translator to complete a large translation task incrementally, by repeatedly downloading content for offline work and then uploading the completed portion at the end of the day. Requiring the entirety of the XLIFF:doc to be translated for a successful response would preclude using the Translate-Strict-Bitext task in this case.
- The per-TUV state stored within an XLIFF:doc is both a more powerful and a more natural place to track translation progress.

"Success" Response

A "Success" response indicates that the request was successfully processed, and that translation activity may have been performed. The XLIFF:doc file should be examined to determine whether the translation is complete.

"Failure" Response

A "Failure" response indicates that the request was not successfully processed, for instance due to:

- Invalid TIPP manifest data, package structure, or security information
- Invalid, corrupt, or missing XLIFF:doc file
- Software error

When possible, the `ResponseComment` element should be used to provide additional information about the failure, such as an error message or stack trace.

Translate-Generic-Bitext

The Translate-Generic-Bitext task type is a variant of the Translate-Strict-Bitext task that trades flexibility in requirements for reduced interoperability. Most notably, a Translate-Generic-Bitext task request can carry more than one bilingual file, and supports bilingual file formats other than XLIFF:doc.

Task ID

The Translate-Generic-Bitext task is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-generic-bitext

Supported Package Object Sections

TIPPs with the Translate-Generic-Bitext task type support the following package sections:

Section	Request TIPP	Response TIPP
Bilingual	Required	Required
Structured Translation Specification	Optional	Optional
TMX	Optional	Optional
Reference	Optional	Optional

Each of these sections is defined in more detail below.

Bilingual

The required *bilingual* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-generic-bitext/bilingual

The bilingual section contains one or more bilingual files for translation. Supported formats include:

- XLIFF:doc
- XLIFF 1.2
- PO (gettext)

The source and target locales for any bilingual files must match the source and target locales in the TIPP manifest.

In the response TIPP, the bilingual section should contain the same files that the request TIPP file made, updated as needed by the translation process.

Structured Transation Specification

The optional *Structured Translation Specification (STS)* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-generic-bitext/sts

This section behaves identically to the corresponding section in the Translate-Strict-Bitext task.

TMX

The optional *TMX* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-generic-bitext/tmx

This section behaves identically to the corresponding section in the Translate-Strict-Bitext task.

Reference

The optional *Reference* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-generic-bitext/reference

This section behaves identically to the corresponding section in the Translate-Strict-Bitext task.

Processing Expectations

As with the Translate-Strict-Bitext task, the "Success" and "Failure" values of a Translate-Generic-Bitext response package do not indicate anything about the extent to which the bilingual files were modified.

"Success" Response

A "Success" response indicates that the request was successfully processed, and that translation activity may have been performed. The individual bilingual files should be examined to determine whether the translation is complete.

"Failure" Response

A "Failure" response indicates that the request was not successfully processed, for instance due to:

- Invalid TIPP manifest data, package structure, or security information
- Invalid, corrupt, or missing bilingual files
- Software error

When possible, the `ResponseComment` element should be used to provide additional information about the failure, such as an error message or stack trace.

Translate-Native-Format

The Translate-Native-Format task type represents a minimal translation operation, in which one or more source files are somehow translated into target files. A situation that might use this task type would be the communication boundary between a Content Management System (CMS) and a translation provider.

Task ID

The Translate-Native-Format task is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-native-format

Supported Package Object Sections

TIPPs with the Translate-Native-Format task type support the following package sections:

Section	Request TIPP	Response TIPP
Input	Required	Optional
Output	Not Present	Required
Structured Translation Specification	Optional	Optional
TMX	Optional	Optional
Reference	Optional	Optional

Each of these sections is defined in more detail below.

Input

The *Input* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-native-format/input

In a request TIPP, the *Input* section must contain one or more source files to be translated. The files should be in the source locale specified in the TIPP manifest. The files may be of any format.

In a response TIPP, the *Input* section is optional and may be omitted.

Output

The *Output* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-native-format/output

In a request TIPP, the *Output* section is not present.

In a response TIPP, the *Output* section must contain a corresponding file for every file that was present in the *Input* section of the request. The files may be:

- Identical to the corresponding *Input* file, for files that were not translated, or were not translatable.
- A translated version of the corresponding *Input* file, for files that were successfully translated.

Structured Transation Specification

The optional *Structured Translation Specification (STS)* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-native-format/sts

This section behaves identically to the corresponding section in the Translate-Strict-Bitext task.

TMX

The optional *TMX* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-native-format/tmx

This section behaves identically to the corresponding section in the Translate-Strict-Bitext task.

Reference

The optional *Reference* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/translate-native-format/reference

This section behaves identically to the corresponding section in the Translate-Strict-Bitext task.

Processing Expectations

As with the Translate-Strict-Bitext task, the "Success" and "Failure" values of a Translate-Native-Format response package do not indicate anything about the extent to which the input files were modified.

"Success" Response

A "Success" response indicates that the request was successfully processed, and that translation activity may have been performed. For each file in the *Input* section in the request TIPP, a corresponding file should be present in the *Output* section in the request TIPP. The individual files should be examined to determine whether the translation is complete.

"Failure" Response

A "Failure" response indicates that the request was not successfully processed, for instance due to:

- Invalid TIPP manifest data, package structure, or security information
- Invalid or corrupt input files
- Software error

When possible, the `ResponseComment` element should be used to provide additional information about the failure, such as an error message or stack trace.

Prepare-Specifications

The Prepare-Specifications task type represents the exchange of information in advance of a translation process, in order to help prepare Structured Translation Specification (STS) files that will advise the translation itself.

For example, the Prepare-Specifications task may be used by a translation customer who is preparing a project for a vendor, with whom they have agreed to use STS to describe the parameters of their translation process. The customer sends sample source files, along with a proposed STS file, to the vendor for review. The response TIPP contains an updated STS file based on the vendor's capabilities.

Note that unlike the Translate tasks, STS data is meant to be consumed and acted on by humans, rather than fully machine-readable. However, there is still value in using TIPP as a standard way to transport this data.

Task ID

The Translate-Native-Format task is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/prepare-specifications

Supported Package Object Sections

TIPPs with the Prepare-Specifications task type support the following package sections:

Section	Request TIPP	Response TIPP
Content	Required	Optional
Structured Translation Specification	Required	Required

Each of these sections is defined in more detail below.

Content

The *Content* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/prepare-specifications/content

In a request TIPP, the *Content* section should contain one or more representative sample source files to be evaluated as part of the STS data collection.

In a response TIPP, the *Content* section is optional and may be

omitted.

Structured Transation Specification

The *Structured Translation Specification (STS)* section is identified by the URI

http://schema.interoperability-now.org/tipp/v1.4/tasks/prepare-specifications/sts

In a request TIPP, the *STS* section should contain an STS file containing any known specifications about the proposed translation process.

In a response TIPP, the STS section should contain an updated version of the STS file.

Processing Expectations

As with the Translate-Strict-Bitext task, the "Success" and "Failure" values of a Translate-Native-Format response package do not indicate anything about the extent to which the input files were modified.

"Success" Response

A "Success" response indicates that the request was successfully processed, and that the STS data was examined and updated. The STS data itself must be examined in order to determine whether additional input is needed.

"Failure" Response

A "Failure" response indicates that the request was not successfully processed, for instance due to:

- Invalid TIPP manifest data, package structure, or security information
- Invalid or corrupt input files
- Software error

When possible, the `ResponseComment` element should be used to provide additional information about the failure, such as an error message or stack trace.

Version specific Information and limitations

Version 1.4.1

This version has the following limitations:

- Encrypted Package Object Containers are not supported. All package Envelopes are expected to contain only an un-

encrypted *pobjects.zip* in addition to the manifest.

- The only type of Package Object that can be included in a TIPP is a file that is directly embedded in the Package Object Container. Future releases may allow references to external releases (for example, reference material accessed via HTTP).

Reference Guide

Naming convention for files

Envelopes should be identified by the suffix **.tipp**. The contents of the Envelope should be named as follows:

Name	Description
`manifest.xml`	TIPP manifest
`pobjects.zip`	Package Object Container (un-encrypted)
`pobjects.zip.enc`	Package Object Container (encrypted)

Naming Restrictions

In order to minimize platform-specific incompatibilities, both Envelopes names and the components of all Package Object paths are restricted to the following subset of ASCII:

- a-z
- A-Z
- 0-9
- Underscore ('_'), dash ('-'), period ('.'), or space (' ')

See also the additional restrictions on path construction described in Format of Package Object Paths.

Tool Identifiers

Information about what tools generate the task and response packages are encoded in the package manifests. In the time prior to the availability of centralized repositories and functionalities tied to tool identity, this information is considered informational. There is currently no mechanism for a task package to require that a particular tool be used to process it and generate the response.

Tools are described by the `Tool` element, and encode three pieces of

data:

- The common name for the tool
- The tool ID, expressed as a URI
- The tool version, expressed as a string

For example, the common name for a tool might be “GlobalSight”, with version “8.1” and ID “http://www.globalsight.com". For now, the specific semantics of Tool IDs are left up to the tool makers.

Communication Endpoint Identifiers

Information about the systems that generate the task and response packages are encoded in the package manifests. There is currently no mechanism for a task package to require that a particular endpoint be used to process it and generate the response.

Communications endpoints are identified by three pieces of information:

- The common name for the endpoint, such as the name of the controlling organization
- The endpoint ID, expressed as a URI, such as the URI of the specific system that generated the package
- A timestamp, recording the time when the package was created according to Format of Date/Time Fields.

The elements used to describe the endpoint differ for task and response packages. Task packages describe the originating endpoint in the `PackageCreator` section, using the `CreatorName`, `CreatorID`, and `CreatorUpdate` fields. Response packages describe the responding endpoint in the `OrderResponse` element, using the `ResponseName`, `ResponseID`, and `ResponseUpdate` fields.

Field	**Originating endpoint**	**Responding endpoint**
Name	`CreatorName`	`ResponseName`
ID	`CreatorID`	`ResponseID`
Timestamp	`CreatorUpdate`	`ResponseUpdate`

Additionally, Communication Endpoint sections include the Tool Identifier of the tool that process the TIPP at that endpoint.

Format of Date/Time Fields

Several *manifest.xml* fields contain date and time data. All of these fields must be contain data formatted according to ISO 8601, using the UTC timezone:

```
YYYY-MM-DDThh:mm:ssZ
```

The format consists of year (4 digits), month (2 digits), day of month (2 digits), the literal string "T", the hour (2 digits), minute (2 digits), and second (2 digits), followed by the literal string "Z" to indicate UTC time.

This format is consistent with XLIFF:doc and the format of the XLIFF date attribute.

Other date/time formats are treated as errors.

Format of Package Object Paths

All `ObjectFile` elements specify the location of an object in the package via their `LocationPath` child element. The value of `LocationPath` must follow the following rules:

- The path is relative to the top-level package folder corresponding to the containing `PackageObjectSection`. The path should not include the name of this folder. For example, for `ObjectFile` elements with an "input" `PackageObjectSection`, all `LocationPath` values are considered relative to the "input" folder in the package, and do not need to be prefixed with "input". Similarly, path values should not be prefixed with "/".
- All paths are considered case-sensitive.

If the package creator creates additional folder structure beneath the top-level package folders, additional rules exist to govern references to objects within these subfolders:

- The forward slash ("/") is used to separate path components.
- All paths must be normalized into a canonical form consisting solely of named path elements and path separators. The path elements "." and ".." are not supported.

In order to maximize cross-platform compatibility, the length of the entire object path, including the section name, must be less than or equal to 240 characters.

References

- RFC 4122, “A Universally Unique IDentifier (UUID) URN Namespace”; http://www.ietf.org/rfc/rfc4122.txt
- ISO 11669, “Structured Specifications and Translation Parameters (version 6.0)”; http://www.ttt.org/specs/
- “XML Signature Syntax and Processing”; http://www.w3.org/TR/xmldsig-core/

TEMP: Misc. Questions

This is for questions we want to track during the development of the reference guide, which are not captured elsewhere in the document.