**The TMS Interoperability Protocol Package (TIPP)**

*V1.5, DRAFT 3, October 26 , 2012*

[Document History](#h.e84d5e98ae51)

[Introduction](#h.5y4a4ta125fn)

[Glossary of Terms](#h.dc2f9eaef0fa)

[Task Types, Requests, and Responses](#h.rfskdgi16pao)

[How TIPP Fits into a Translation Supply Chain](#h.2oggcs4ax27b)

[TIPP Structure](#h.de8ccd0ea4b7)

[Success and Failure](#h.66xafwk3qwlu)

[Package Object Sections](#h.gbo9cd305jk6)

[Standard Task Types](#h.6xayb8ty57xo)

[Developing Custom Task Types](#h.83fubv86yvs0)

[Compliance](#h.m0cl0957oyg3)

[Multilingual TIP Packages (mTIPP)](#h.gt11psciyxf8)

[Reference Guide](#h.e5580bc918cf)

[References](#h.9k8yhbc12o0r)

# 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. |
| 9 | 1.5.0 Draft 1 | 7/12 | Chase Tingley | Revisions based on discussions with Linport working group and early implementers. Edits from Jörg Schütz. |
| 10 | 1.5.0 Draft 2 | 3/9/12 | Chase Tingley | Additional revisions from group feedback: failure cases, metrics data, terminology. |
| 10 | 1.5.0 Draft 3 | 26/10/12 | Chase Tingley | Added information about security and multi-target packages. |

# Introduction

The Translation Interoperability Protocol Package (TIPP) is a container format that allows the exchange of information between different points in a translation or localization supply chain.

This reference guide describes the package and standard methods to interact with the package.

## Why TIPP?

As the translation industry matures, the supply chains involved in the process of translation are becoming increasingly complex. During the translation lifecycle of a single translation unit, multiple tools controlled by different parties may be involved.

During translation or localization, source material 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 over the course of a translation or localization lifecycle, in addition to tools such as workbenches or automated quality checkers.

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 localizable and translatable content as well as additional process-specific metadata.
* Standardized 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.

While these solutions are effective within limited environments, neither is currently sufficient for general-purpose, cross-system use. Proprietary solutions are often poorly-documented and too specialized to be deployed in the heterogenous IT environments that characterize today’s automated translation industry landscapes. Existing standards, while an important component to an interoperable solution, are only solving parts of the overall problem.

## What is TIPP?

TIPP is a means to transport localization- and translation-related data and metadata.

As such, TIPP can be seen as a replacement of the ad-hoc system of FTP sites, zip files, and impromptu packaging formats that inhabit the traditional 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 flexiblity 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 specification defines several built-in tasks, representing different core behaviors in the translation supply chain. 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 type, whether it is a request or a response, and certain types of generic metadata that are common across all task types. It also enumerates the contents of the TIPP payload. It also contains security information related to the digital signing of the payload.
* **Payload**. Additional data in a TIPP is known as the payload. The payload contains user data, tool metadata, etc. Different task types allow for different types of payload data. The payload is stored in an internal container within the TIPP, and is optionally digitally signed.

## What are the 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 translation tools.
* When used in conjunction with other bilingual formats such as XLIFF 1.2 or gettext (.po), a lesser guarantee of interoperability between translation tools.
* In the absence of well-defined bilingual formats, the ability to function as a basic packaging mechanism between TMS, between a TMS and a CMS, or between other tools related to translation and localization.

## 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 discover the package contents and to encode certain pieces of metadata that are useful regardless of task type. The metadata includes:
  + Task type information, including information related to the request/response behavior.
  + Information that identifies the package creator
  + A unique package identifier
* **Task types impose their own requirements.** TIPP task types may mandate additional restrictions on the resources that they may carry as a payload, as well as the processing expectations for handling a TIPP request of that task type. The highest level of interoperability can be achieved by using the standard [Translate-Strict-Bitext](#id.iryn6ntmyd3k) task type, while the [Translate-Generic-Bitext](#id.pgfupizdvecb) and [Translate-Native-Format](#id.jg7or7d2nyx) task types allow for fewer usage constraints in exchange for less guaranteed interoperability.

## Is TIPP enough to guarantee interoperability on its own?

No. TIPP is first and foremost a standard way to represent certain types of data for exchange between tools. Other types of data involved in a translation supply chain may also need standardized representations, as well as well-defined rules for how the data should be interpreted and handled.

However, the Interoperability Now! working group has developed TIPP to work best in conjunction with other commonly agreed on or commonly adopted formats, such as XLIFF:doc.

## Task Types

The TIPP specification defines 4 standard task types:

* [Translate-Strict-Bitext](#id.iryn6ntmyd3k)
* [Translate-Generic-Bitext](#id.pgfupizdvecb)
* [Translate-Native-Format](#id.jg7or7d2nyx)
* [Prepare-Specifications](#id.ngbr6p9rptm1)

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

Custom Task Types may be defined to handle additional use cases. Information on Custom Task Types can be found in the [Developing Custom Tasks](#id.i51kow55s8vd) section.

## Versioning of this Reference

TIPP is being developed using an iterative approach, and will change over time based on implementation experiences. The version number of this document will be incremented over time as an indicator of what has changed.

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 indicate substantial conceptual changes.

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 a ZIP archive located within the Envelope. |
| Payload | Alternate name for Package Object Container. |
| Object | An individual payload item in a TIPP. In version 1.5, these are always files contained in the Payload. Future TIPP versions may allow Objects to be referenced externally (eg, via HTTP). |
| Request | A TIPP that defines a particular localization or translation task to be completed. |
| Resource | Alternate name for Object. |
| Response | A TIPP that contain the results of an attempt to complete the localization or translation 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 standard task types, but others can be defined by third parties. |

# Task Types, Requests, and Responses

Every TIPP represents 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 translation or localization process. A request TIPP specifies the task type and the resources required to perform the 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-Strict-Bitext](#id.iryn6ntmyd3k) task type defines the translation of a single XLIFF:doc file. A request TIPP with the [Translate-Strict-Bitext](#id.iryn6ntmyd3k) task type contains:

* Metadata that is common to all TIPPs, such as source and target locale and information about the party that generated the request.
* 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

After processing a request TIPP, a response TIPP with the [Translate-Strict-Bitext](#id.iryn6ntmyd3k) task type is generated. The response TIPP contains:

* Metadata identifying the original request.
* Metadata related to the response, including a success indicator and information about the party that generated the response.
* An updated version of the XLIFF:doc from the request, containing the results of any translation performed.

The standard task types are defined in [Standard Tasks](#id.mqst91sb2jjj).

# How TIPP Fits into a Translation Supply Chain

This example demonstrates how a series of TIPP request/response pairs can be used to implement data exchange at different points in a translation workflow. Additionally, it shows how a combination of task types with different guarantees of interoperability can be used to extend the use of TIPP for purposes other than communication between TMSs. In this case, the use of TIPP in a workflow extends to a customer CMS as well.



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 [Translate-Native-Format](#id.jg7or7d2nyx) 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-Strict-Bitext](#id.iryn6ntmyd3k) 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-Strict-Bitext](#id.iryn6ntmyd3k) 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-Strict-Bitext](#id.iryn6ntmyd3k) 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-Strict-Bitext](#id.iryn6ntmyd3k)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 [Translate-Native-Format](#id.jg7or7d2nyx) 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 [Translate-Generic-Bitext](#id.pgfupizdvecb) 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.
* A 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 *PackageObjects.zip.*

## TIPP Manifest (manifest.xml)

The manifest is an XML document conforming to the schema located at

http://schema.interoperability-now.org/tipp/TIPPManifest-1\_5.xsd

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

### Package ID

Each TIPP is assigned 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](#id.re0z5zewy93g).
* *Tool Identifier*, describing the tool that created the package. Each tool should be identified as specified in [Tool Identifiers](#id.ugssocr37445).

### Task Information

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

A response TIPP includes 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](#id.5xbvj8wmypo6).
* A response message, either “Success” or “Failure”, depending on whether the task was executed successfully. Individual task types define their own criteria for returning “Success”.
* 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 enumerates all the Package Objects contained in the TIPP. A TIPP that contains Objects in the payload that is not described by the manifest must be considered invalid.

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. The purpose of the name is to allow implementations to map the section to a specific location within the payload’s zip archive.

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.

#### Object Name vs Object Path

Package Objects are described in the manifest with both a path and a path. While in some cases these two values may be identical, it is important that they be treated separately.

The name represents an abstract identifier for the resource. The name may be arbitrarily long and contain any Unicode character. The name may have meaning outside the TIPP; for example, it may be a user-readable file name, or a location within a resource hierarchy in a TMS. TIPP implementations must not modify Object names during processing.

In contrast, the path is an implementation detail internal to TIPP. Paths identify specific locations within the TIPP itself. In the current implementation, the path represents a location within the payload ZIP archive. Paths are subject to restrictions defined both by the ZIP format and by the file systems on which it is expected a TIPP may be manipulated. The format of paths is described in [Format of Package Object Paths](#id.a94cc8725542).

Paths are created when a given TIPP is assembled and have no meaning beyond the scope of that TIPP. A single Object that is included as part of a request TIPP and a subsequent response TIPP is expected to have a constant name, but may be assigned a different path for each TIPP that carries it. An implementation must not assume that a particular path will be assigned to a certain Object, even if it has been assigned to that object in a previous TIPP.

### Security

TIPPs support optional digital signatures via XML-DSIG. The TIPP schema includes an optional XML-DSIG <Signature> element as part of the manifest. If this element is present, it should contain a valid digital signature covering the entirety of the package.

An XML-DSIG signature should contain two references to objects covered by the signature:

* The manifest, identified by the reserved URI=”” value. As the manifest contains the signature itself, this is what XML-DSIG refers to as an “enveloped” signature.
* The payload, identified by name. The signature is computed over the bytes of the compressed payload ZIP, rather than over the individual package objects.

## Package Object Container (PackageObjects.zip)

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 1.5, the Package Object Container is represented as an embedded ZIP archive called *PackageObjects..zip*.

### 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](#id.a94cc8725542). 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.

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

# Success and Failure

TIPP responses have a limited ability to describe the results of processing their corresponding task request. In the <ResponseMessage> element, a TIPP response may indicate that the task was completed successfully, or it may report one of several failure codes. Additional, human-readable details may be provided in the <ResponseComment> element.

The set of failure cases that a TIPP response can report does not cover all possible failures. For example, if a TIPP request contains a corrupt or unparseable manifest, it may not be possible to construct a valid TIPP response to it to report the error. In order to fully cover all possible error cases, the transmission of TIPP requests and responses would need to be ensured by an additional delivery protocol. Such a protocol is currently beyond the scope of this specification.

It is expected that most failure cases will not be correctable without human intervention. When an error occurs while processing a TIPP, it is the responsibility of each tool to provide information about the error to the appropriate user, so that the user may take appropriate action, including potentially contacting the party that generated the faulty TIPP.

The following values for the <ResponseMessage> element are supported:

* Success: Indicates that the TIPP request was processed and the task completed successfully. The specific conditions under which a task response may return the Success are delegated to the individual task types.
* Invalid Manifest: Indicates that the manifest TIPP request contained an error that prevents the tasks from being completed. As noted above, this does not cover errors that prevent the manifest itself from being parsed. Conditions that would result in an Invalid Manifest message include:
  + The payload contains resources that are not listed in the manifest.
  + The manifest contains resources that are not present in the payload.
  + One or more sections present in the manifest are not allowed for the specified task type.
* Invalid Payload: Indicates that there was a problem with the package payload that prevents the task from being completed successfully. Conditions that would result in an Invalid Payload message include:
  + The payload can not be opened or is otherwise corrupt.
  + One or more resources that are required to complete the task are missing.
* Security Failure: Indicates that there was a problem with the package’s digital signature, if one was present. Conditions that would result in a Security Failure message include:
  + The digital signature found in the manifest could not be verified.
  + The digital signature was present, but did not cover both the manifest and the payload.
* Unsupported Task Type: Indicates that the TIPP request specified a task type that the recipient does not support.
* Task Failure: Indicates that the TIPP request could not be completed successfully for reasons that are specific to the task type.

Task types must specify the criteria for reporting Success in a response TIPP of that task type. Task types may also specify additional criteria for reporting Task Failure in a response TIPP of that task type.

# Package Object Sections

This section enumerates the Package Object Sections allowed by TIPP, along with a corresponding URI and usage overview for each. Each task type makes use of one or more of the Package Object Sections and must define its specific semantics within the context of that task type.

## Bilingual

http://schema.interoperability-now.org/tipp/v1.5/sections/bilingual

The Bilingual section contains bilingual resources, such as XLIFF:doc files. Task types that use this section should further specify what restrictions exist on its resources.

## Input

http://schema.interoperability-now.org/tipp/v1.5/sections/input

The Input section contains resources that represent the input to some translation or localization business process. Task types that use this section should further specify what restrictions exist on its resources.

Task types that use the Input section may also use the Output section, but are not required to do so.

## Output

http://schema.interoperability-now.org/tipp/v1.5/sections/output

The Output section contains resources that represent the output to some translation or localization business process. Task types that use this section should further specify what restrictions exist on its resources.

Task types that use the Output section may also use the Input section, but are not required to do so.

## Structured Translation Specification

http://schema.interoperability-now.org/tipp/v1.5/sections/sts

The Structured Translation Specification section contains resources in the format defined by the forthcoming ISO 11669.

## TM

http://schema.interoperability-now.org/tipp/v1.5/sections/tm

The TM section contains translation memory resources. Task types that use this section should further specify what restrictions exist on its resources.

## Terminology

http://schema.interoperability-now.org/tipp/v1.5/sections/terminology

The Terminology section contains terminology resources. Task types that use this section should further specify what restrictions exist on its resources.

## Reference

http://schema.interoperability-now.org/tipp/v1.5/sections/reference

[T](http://interoperability-now.googlecode.com/schema/tasks/v1/translate/bilingual)he Reference section contains resources that serve as reference for a translation or localization business process. Task types that use this section should further specify what restrictions exist on its resources.

## Preview

http://schema.interoperability-now.org/tipp/v1.5/sections/preview

The Preview section contains resources that are are associated with an automated context preview operation as part of a translation or localization step. Unlike the [Reference](#id.kx8obkbpkc8t) section, which frequently contains resources intended for humans to read, resource in the Preview section are intended to be support files for an automated process.

## Metrics

http://schema.interoperability-now.org/tipp/v1.5/sections/metrics

The Metrics section contains resources that provide word counts, character counts, or other metrics related to the task. Task types that use this section should further specify what restrictions exist on its resources.

## Custom

http://schema.interoperability-now.org/tipp/v1.5/sections/custom

[T](http://interoperability-now.googlecode.com/schema/tasks/v1/translate/bilingual)he Custom section contains resources that are not appropriate for any other section. Use of the Custom section is reserved for Custom Task Types. Task types that use this section should further specify what restrictions exist on its resources.

# Standard Task Types

This section enumerates the TIPP Tasks defined as part of the Interoperability Now! effort.

## 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](#id.pgfupizdvecb) task type.

### Task ID

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

http://schema.interoperability-now.org/tipp/v1.5/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](#id.ndjtbhg9zvws) | Required | Required |
| [Structured Translation Specification](#id.9x49dut5ewqx) | Optional | Optional |
| [Preview](#id.ppxy9pnn4ysw) | Optional | Optional |
| [TM](#id.m24bw4cudw8n) | Optional | Optional |
| [Metrics](#id.87ozo8f87f44) | Optional | Optional |
| [Reference](#id.kx8obkbpkc8t) | Optional | Optional |

The use of each of these sections is described in more detail below.

#### 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

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

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

#### TM

This optional section is used by a request TIPP to carry supplemental translation memory information to be used during the translation process. The TM section may contain one or more resources, 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.

#### Metrics

#### This optional section contains a GMX-V file that provides metrics data related to the translatable portion of the payload. The GMX-V file must use the <project> element to specify count information for each resource in the Bilingual section of the payload. TIPP does not restrict what GMX-V counts can be used as part of the provided metrics data.

#### Preview

This optional section is used by a request TIPP to carry supplemental resources used in the preview process provided by the XLIFF:doc file.

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

#### Reference

This optional section is used by a request TIPP to carry supplemental resources that may be of use during the translation process. Examples include source files, style guides, or other notes from the content creator. The Translate-Strict-Bitext task assigns no particular meaning to these resources 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 its XLIFF:doc payload to be processed in accordance with the XLIFF:doc specification, and to receive the processed XLIFF:doc as part of a successful Translate-Strict-Bitext response. However, it is not a required 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 common 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. Each of these daily cycles is best represented as a separate TIPP request/response pair.
* 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 some translation activity may have been performed. The state within the XLIFF:doc itself should be used to determine the extent to which translation was completed.

#### “Failure” Response

In addition to standard reasons for failure identified in [Success and Failure](#id.fikh8cv4758d), a “Failure” response should also be provided when the XLIFF:doc resource in the payload is missing or invalid. Additional information about the failure may be provided in the Comment field of the manifest.

## Translate-Generic-Bitext

The Translate-Generic-Bitext task type is a variant of the [Translate-Strict-Bitext](#id.iryn6ntmyd3k) 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/v1.5/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](#id.ndjtbhg9zvws) | Required | Required |
| [Structured Translation Specification](#id.9x49dut5ewqx) | Optional | Optional |
| [TM](#id.m24bw4cudw8n) | Optional | Optional |
| [Terminology](#id.ldizlztj47r) | Optional | Optional |
| [Metrics](#id.87ozo8f87f44) | Optional | Optional |
| [Reference](#id.kx8obkbpkc8t) | Optional | Optional |

The use of each of these sections is described in more detail below.

#### Bilingual

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

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

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

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

#### Structured Transation Specification

This optional section behaves identically to the corresponding section in the [Translate-Strict-Bitext](#id.iryn6ntmyd3k) task.

#### TM

This optional section behaves identically to the corresponding section in the [Translate-Strict-Bitext](#id.iryn6ntmyd3k) task.

#### Terminology

This optional section contains terminology data related to the translatable portion of the payload. Terminology resources must be formatted according to either the UTX or TBX-Glossary specifications.

#### Metrics

#### This optional section contains a GMX-V file that provides metrics data related to the translatable portion of the payload. The GMX-V file must use the <project> element to specify count information for each resource in the Bilingual section of the payload. TIPP does not restrict what GMX-V counts can be used as part of the provided metrics data.

#### Reference

This optional section behaves identically to the corresponding section in the [Translate-Strict-Bitext](#id.iryn6ntmyd3k) task.

### Processing Expectations

As with the [Translate-Strict-Bitext](#id.iryn6ntmyd3k) task, the “Success” and “Failure” values of a Translate-Generic-Bitext response package do not indicate anything about the extent to which the bilingual resources 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 resources should be examined to determine whether the translation is complete.

#### “Failure” Response

In addition to standard reasons for failure identified in [Success and Failure](#id.fikh8cv4758d), a “Failure” response should also be provided when the bilingual resource in the payload is missing or invalid. Additional information about the failure may be provided in the Comment field of the manifest.

## Translate-Native-Format

The Translate-Native-Format task type represents an abstract translation process by which one or more source resources are translated into target resources. No additional assumptions are made about the format of either the source or target resource. A situation that might use this task type would be the communication boundary between a Content Management System (CMS) and a translation provider, such as the example provided in [How TIPP Fits into a Translation Supply Chain](#id.ggzvlugtpnx6).

### Task ID

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

http://schema.interoperability-now.org/tipp/v1.5/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](#id.4fcl7fmrvuap) | Required | Optional |
| [Output](#id.a2goejjzeyhf) | Not Present | Required |
| [Structured Translation Specification](#id.9x49dut5ewqx) | Optional | Optional |
| [TM](#id.m24bw4cudw8n) | Optional | Optional |
| [Terminology](#id.ldizlztj47r) | Optional | Optional |
| [Metrics](#id.87ozo8f87f44) | Optional | Optional |
| [Reference](#id.kx8obkbpkc8t) | Optional | Optional |

The use of each of these sections is described in more detail below.

#### Input

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

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

#### 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 resources may be:

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

#### Structured Transation Specification

This optional section behaves identically to the corresponding section in the [Translate-Strict-Bitext](#id.iryn6ntmyd3k) task.

#### TM

This optional section behaves identically to the corresponding section in the [Translate-Strict-Bitext](#id.iryn6ntmyd3k) task.

#### Terminology

This optional section behaves identically to the corresponding section in the [Translate-Generic-Bitext](#id.pgfupizdvecb) task.

#### Metrics

#### This optional section contains a GMX-V file that provides metrics data related to the translatable portion of the payload. The GMX-V file must use the <project> element to specify count information for each resource in the Input section of the payload. TIPP does not restrict what GMX-V counts can be used as part of the provided metrics data.

#### Reference

This optional section behaves identically to the corresponding section in the [Translate-Strict-Bitext](#id.iryn6ntmyd3k) task.

### Processing Expectations

As with the [Translate-Strict-Bitext](#id.iryn6ntmyd3k) task, the “Success” and “Failure” values of a Translate-Native-Format response package do not indicate anything about the extent to which the input resources 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 resources should be examined to determine whether the translation is complete.

#### “Failure” Response

In addition to standard reasons for failure identified in [Success and Failure](#id.fikh8cv4758d), a “Failure” response should also be provided when no translation could be performed for reasons related to problems with the request TIPP, such as missing or invalid input resources. Additional information about the failure may be provided in the Comment field of the manifest.

## 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) resources 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 resources, 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/v1.5/tasks/prepare-specifications

### Supported Package Object Sections

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

|  |  |  |
| --- | --- | --- |
| **Section** | **Request TIPP** | **Response TIPP** |
| [Input](#id.4fcl7fmrvuap) | Required | Optional |
| [Structured Translation Specification](#id.9x49dut5ewqx) | Required | Required |

The use of each of these sections is described in more detail below.

#### Input

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

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

#### Structured Transation Specification

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](#id.iryn6ntmyd3k) task, the “Success” and “Failure” values of a Translate-Native-Format response package do not indicate anything about the extent to which the input resources 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

In addition to standard reasons for failure identified in [Success and Failure](#id.fikh8cv4758d), a “Failure” response should also be provided when the STS resource is missing or invalid. Additional information about the failure may be provided in the Comment field of the manifest..

# Developing Custom Task Types

Custom Task Types may be defined to handle use cases other than those identified by this specification. 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 must include:

* A URI that uniquely identifies the custom task type.
* The Package Object Sections supported by Request and Response TIPPs of the custom task type.
* For each supported 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.

While there is no required format for a custom task type definition, it is proposed that the format used in [Standard Task Types](#id.mqst91sb2jjj) be used as a model.

## Use of the Custom Package Object Section

The [Custom](#id.7jp1vghk6ibv) Package Object Section is reserved solely for use in Custom Task Types. It exists to allow Custom Task Types to carry resources that do not logically fit within another Package Object Section.

# Compliance

Regardless of task type, compliant TIPP implementations are required to:

* Read and write valid TIPPs, as describe by this document.
* Read and write valid TIPP manifests, as prescribed by this document and the TIPPManifest-1\_5.xsd XML schema.

Evaluating compliance with a particular TIPP task type depends on the task type itself. Task types offer two levels of TIPP compliance:

1. **Semantic (Strict) Interoperability** - Any compliant process or service must read and consume the TIPP in the same way, without need for additional information about the TIPP. Data loss shall not occur. Semantic interoperability should be verified via automatic tests that are specific to the task type. Task types that offer semantic interoperability should include the identifier “strict” in the task name. For example, the [Translate-Strict-Bitext](#id.iryn6ntmyd3k) task type offers Semantic Interoperability.
2. **Syntactic (Generic) Interoperability** - Any compliant process or service should know about the task type, should know where to find the resources associated with the task, and should be able to read and write the TIPP request/response. Additional information, not contained in the TIPP, may be required to enable tools to process the data in the TIPP correctly. This type of interoperability cannot be verified via automatic tests generic to the task type, and data integrity can not be guaranteed. Tasks of this type should not include the identifier "strict" in the task name. For example, the [Translate-Generic-Bitext](#id.pgfupizdvecb) and [Translate-Native-Format](#id.jg7or7d2nyx) task types offer Syntactic Interoperability.

# Multilingual TIP Packages (mTIPP)

TIPPs are always bilingual, specifying a single source and a single target language. However, translation and localization processes often involve multiple target languages, and sometimes multiple source languages as well. mTIPP is a simple extension to TIPP to allow for multiple bilingual TIPPs to be gathered together in a multilingual wrapper.

An mTIPP can contain any number of TIPPs, in any combination of source and target language pairs.

## mTIPP Layout

An mTIPP is implemented as a ZIP archive, containing:

* The mTIPP descriptor - an XML file conforming to the schema

http://schema.interoperability-now.org/tipp/TIPPMultiWrapper-1\_0.xsd

* One or more TIPPs.

The descriptor enumerates the TIPPs found in the mTIPP and identifies the source and target language for each. Note that multiple TIPPs may be present for a single source/target language pair.

## Structure of the mTIPP Descriptor

An mTIPP descriptor consists of one or more <Packages> element, each identifying a source language. Each <Packages> element contains one or more <Package> elements, each identifying a target language and a TIPP with that source/target language pair.

# 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 |
| PackageObjects.zip | Package Object Container |

### 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.](#id.a94cc8725542)

## 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](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.](#id.e1bed2c908ff)

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

* Bly, Micah; “Xliff:doc Representation Guide v.0952”; <http://interoperability-now.org>
* Zydroń, Andrzej, et. al.; “Global Information Management Metrics Volume (GMX-V) 2.0 Specification”; <http://docbox.etsi.org/ISG/Open/ISGLIS/GMX-V/GMX-V/GMX-V-2.0.html>
* RFC 4122; “A Universally Unique IDentifier (UUID) URN Namespace”; [http://www.ietf.org/rfc/rfc4122.txt](http://www.google.com/url?q=http%3A%2F%2Fwww.ietf.org%2Frfc%2Frfc4122.txt&sa=D&sntz=1&usg=AFQjCNH8HAZqCWFXW6hESL3pZUvInoq8Nw)
* ISO 11669; “Structured Specifications and Translation Parameters (version 6.0)”; <http://www.ttt.org/specs/>
* “Universal Terminology Exchange (UTX)”; <http://www.aamt.info/english/utx/index.htm>
* “TBX-Glossary”; <http://www.ttt.org/tbxg/>
* “XML Signature Syntax and Processing”; <http://www.w3.org/TR/xmldsig-core/>