DRAFT: The Translation Interoperability Protocol Package (TIPP)

Version 2.0 (draft), April 23, 2019

Version 2.0 by the Linport working group within Translation Commons
based on version 1.5 by Interoperability Now!

Mikel L. Forcada, European Association for Machine Translation, president
Andrzej Zydroń, XTM Intl., CTO
Alan K. Melby, LTAC Global, president
Marc Mittag, MittagQI, CEO
Chase Tingley, Interoperability Now!
Jeannette Stewart, Translation Commons, founder

Note: We anticipate collaboration with TAPICC, specially working group 2 (payload). TIPP does not specify an API. All references to files and ZIP containers in this document should be understood as being *virtual* files or object models. We hope to harmonize the TIPP object model and the TAPICC payload model. Any suitable API should be capable of creating these files.

1 Introduction

The Translation Interoperability Protocol Package[1] (TIPP) specification defines a container format that allows the exchange of information between different points in a translation or localization supply chain.

The TIPP 1.5 standard was created by the Interoperability Now! committee and published in December 2015. For details please refer to https://github.com/tingley/interoperability-now The work of the Interoperability Now! committee has been taken over by the Linport working group under the auspices of Translation Commons with a goal of updating the original TIPP standard to reflect recent changes to Localization Standards and also leverage practical experience regarding TIPP implementations over the past 5 years.

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

1.1 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 modified by multiple tools controlled by several different parties. Where once translation management system (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 2 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.

[1] Formerly “TMS Interoperability Protocol Package”, with TMS standing for *translation management system*.

Existing standards, while an important component to an interoperable solution, are only solving parts of the overall problem.

1.2 What is TIPP?

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

As such, a TIPP based process 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. The TIPP format 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 flexibility 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 2 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, an internal contained within the TIPP.

Resources. Data necessary to perform a task is provided in the form of resources. The resources represent user data, tool metadata, etc. Different task types use different types of resources. Resources in a TIPP are stored inside the payload.

1.3 What are the goals of the TIPP specification?

The TIPP specification 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 2, a strong guarantee of interoperability between translation tools. The issue of XLIFF 2 modules will be addressed in detail below.

In the early stages of a translation project, to facilitate the exchange of Structured Translation Specification (STS) information.

1.4 How does the TIPP specification 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:

i. Task type information, including information related to the request/response behavior.
ii. Information that identifies the TIPP creator
iii. 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.

1.5 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 Linport working group within Translation Commons has developed TIPP to work best in conjunction with other commonly agreed on or commonly adopted formats, such as XLIFF 2 (http://docs.oasis-open.org/xliff/xliff-core/v2.0/csprd01/xliff-core-v2.0-csprd01.html)
.

1.6 Task Types

The TIPP specification defines 2 standard task types:

Translate-Strict-Bitext

Prepare-Specifications

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 Task Types section.

1.7 Versioning of this Reference

The TIPP specification 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.*

2 Glossary of Terms

Envelope	The outermost container in a TIPP. In the current implementation, this is a ZIP archive as per ISO/IEC 21320-1:2015.
Manifest	An XML file that contains all metadata needed to process a TIPP.
Payload	The portion of a TIPP that stores the resources. The Payload is represented as a folder within the Envelope.
Resource	An individual data object necessary for completion of the task described by a TIPP. In version 2.0, these are always files contained in the Payload. It is intended for the next version to map this also to data objects.
Request	A TIPP that defines a particular localization or translation task to be completed.
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.

3 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. The response TIPP contains the results of an attempt to process the request, and optionally, some or all of the resources that have been produced in the course of processing the request.

For example, the Translate-Strict-Bitext task type defines the translation of one or more single XLIFF 2 files. A request TIPP with the Translate-Strict-Bitext 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.

One or more XLIFF 2 files containing content to be translated for the language pair of the TIPP..

Supplemental resources associated with translation, possibly including (for more information please see “sections”):

A Structured Translation Specification (STS) file

A TMX file with additional translation memory matches

Terminology data

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 task type is generated. The response TIPP contains:

Metadata identifying the original request.

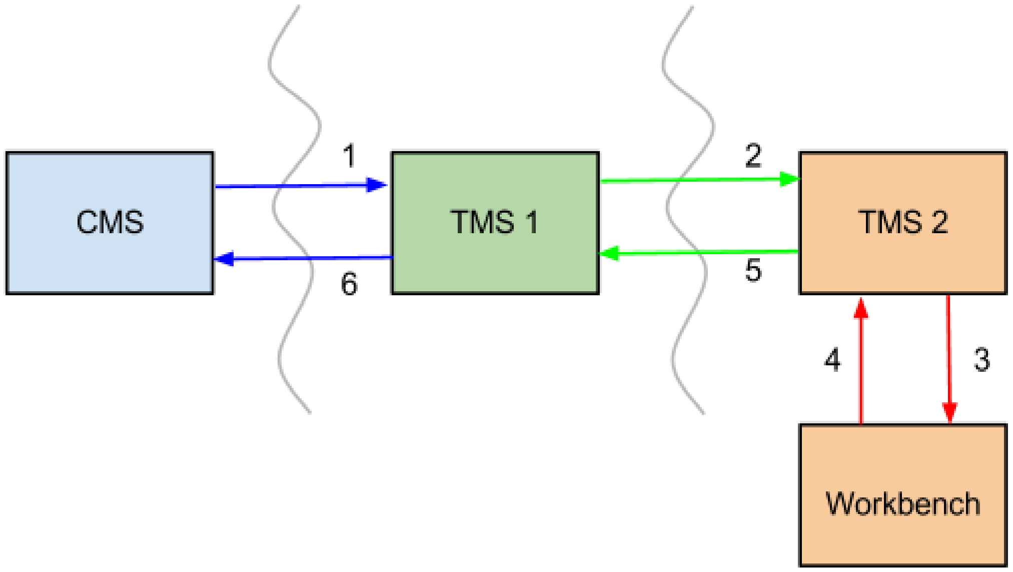
Metadata related to the response, including a success indicator and information about the party that generated the response.

An updated version of the XLIFF 2 files from the request, containing the results of any translation performed.

The standard task types are defined in Standard Task Types.

4 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 Content Management System (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 of the customer exports XLIFF 2 files and generates a TIPP request with the Translate-Strict-Bitext 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 task type, containing an XLIFF 2 file with content received from the customer. The TIPP

request is sent to the SLV's TMS, where the XLIFF 2 is processed through the SLV's translation workflow.

3 The SLV's TMS creates another TIPP request with the Translate-Strict-Bitext task type, again containing the XLIFF 2 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 task type, containing an updated copy of the XLIFF 2 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 task type, containing the translated XLIFF 2 file. The TIPP response is sent back to the MLV's TMS, which completes its translation workflow.

6 The MLV's TMS generates a TIPP response with the Translate-Strict-Bitext task type. The TIPP response is sent back to the customers CMS. The CMS processes the contained XLIFF files and parses the content back into its database.

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.

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, the TIPP specification 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 or other means like a Linport portfolio.

5 TIPP Structure

A TIPP consists of two parts:

- The manifest, an XML file defined by the TIPP Schema.
- The payload, a folder containing one or more resources.

These two parts are contained in a ZIP archive called the Envelope.

The manifest must be named *manifest.xml* and the payload is called *resources.*

5.1 TIPP Manifest (manifest.xml)

The manifest is an XML document conforming to the schema identified by the URI

http://schema.interoperability-now.org/tipp/2_0/TIPPManifest.xsd

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

5.1.1 Package ID

Each TIPP is assigned a 128-bit Universally Unique Identifier (UUID), encoded as a URI, as described in RFC 4122.

5.1.2 Creator Information

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

- **Name**: the name of the company or individual who created the TIPP.
- **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.
- **Date**: the time at which the package was created, as specified in Format of Date/Time Fields.[2]
- **Tool**: an identifier describing the tool that created the package, as specified in Tool Identifiers.

5.1.3 Task Information

Manifests for both request and response TIPPs include a Task descriptor, containing:

- URI: Containing the URI to the task type
- Modules: Required modules for the task type; each module is listed in a sub element module (optional; each task type defines, if it allows modules and if yes lists the allowed modules). Each task type defines separately the modules it allows - if any.

  Listed modules have the required attribute “required” with the values “yes” or “no”. If a tool parses a TIPP package with listed modules, that it does not support and the module is NOT listed as optional in the manifest, it must respond with a response failure of the type “task failure”. The unsupported module must be listed in the comment field of the manifest.

  If a tool parses a TIPP with listed modules, that it does not support and the module is listed as optional in the manifest, it must process the TIPP and ignore the module content.
- Source language identifier, as described in Language Identifiers.

[2] Note that the Package ID is a UUID and therefore contains a bit-embedded timestamp as a means to generate unique identifiers. This timestamp will be ignored. Timestamping information for the TIPP will be that in the Date section of the Creator information in the manifest.

- Target language identifier, as described in Tool Identifiers.
- Terminology precedence: Task types may contain a glossary in special parts of the bilingual files, like in the XLIFF 2 glossary module.
  Allowed values are “bitext” and “terminology”. If the glossary in the bilingual files takes precedence over a glossary in the terminology section of the TIPP, the value must be set to “bitext”. In case the terminology section takes precedence, the value must be set to “terminology”.
  The definition in the manifest.xml is optional; if omitted it defaults to “bitext”.

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 Creator Information.
- A response value, and optionally a human-readable comment, as described in Success and Failure.

5.1.4 Resource Information

The manifest enumerates all the resources contained in the TIPP. A TIPP that contains resources in the payload that are not described by the manifest must be considered invalid.

The payload contains zero or more sections, each of which contains zero or more resources. Each section contains resources for a specific purpose. The available sections are described in Sections.

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

Several pieces of information are provided about each resource:

- The path, which identifies the location of the resource data. In the current implementation of TIPP, all resources are files contained within the payload. Accordingly, the path is a file path within the TIPP payload relative to the the payload folder *resources*.
- The resource name, which is a human-readable identifier. If no name is provided, the resource is identified by its path.
- The sequence number, which is a non-zero positive integer. Sequence numbers provide a way for the TIPP package creator to order the resources within a given section in order to provide additional contextual information.

5.1.4.1 Resource Name vs Location

Resources are described in the manifest with both a location and an optional name. 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 package; 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 resource names during processing.

In contrast, the path is an implementation detail internal to TIPP packages. The structure of the path may be constrained by the type of resource it identifies. In the current implementation, the path represents a location within the payload. Paths are subject to restrictions defined both by the ZIP format and by the file systems on which it is expected a TIPP package may be manipulated. The format of paths is described in Format of Resource Paths.

Paths are created when a given TIPP is assembled and have no meaning beyond the scope of that TIPP. A single resource 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 resource, even if it has been assigned to that object in a previous TIPP.

5.2 Payload

The payload contains zero or more resource sections, each of which contains zero or more resources. The structure of the payload corresponds to the structure defined in the manifest, with each section corresponding to a directory in the payload with a section-specific name.

If the payload is empty, it may be omitted from the TIPP package.

The payload is represented as a folder called *resources*.

5.2.1 Resource Sections

Resource sections are represented by directories within the *resources* folder. Each directory is identified by a name specified by its section type. There can only be one instance of a given section in a payload.

Resources in a section are represented as files within the payload, in the appropriate directory for that section. Additional subdirectories may be created within the top-level section directories.

All resources within the payload must follow the path and naming restrictions described in Format of Resource Paths. As a result, implementations may need to abbreviate or normalize resource names in order to generate paths within the package.

The original name for a resource should be stored in the Name field for that resource in the manifest.

Each task type definition enumerates the allowed sections for that task type from the set of available sections described in Sections.

In any TIPP package, a section that contains no resources may be omitted from the payload.

6 Success and Failure

TIPP responses have a limited ability to describe the results of processing their corresponding task request. In the `<Response>` element, a TIPP response may indicate that the task was completed successfully, or it may report one of several failure codes.

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, as information to identify the faulty request TIPP will not be available. 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 beyond the scope of this document.

It is expected that many 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 `<Response>` 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 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't be opened or is otherwise corrupt.

- o The payload contains resources that are not listed in the manifest.
- o One or more resources that are required to complete the task are missing.

`Unsupported Task Type`: Indicates that the TIPP request specified a task type that the recipient does not support. In this case the payload is returned unchanged in the response package.

`Unsupported XLIFF 2 Module xxxx`: Indicates that the TIPP request requires a XLIFF 2 module for its task type (xxxx), 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.

6.1 Use of the Comment Field

Additional, human-readable details may be provided in the `<Comment>` element. The use of this field is left up to the implementation, but implementers are encouraged to use it to provide relevant information that may help correct any errors that to unsuccessful processing of a TIPP request.

For example, if manifest in request TIPP references several files that are missing from the payload, the TIPP response should use the `Invalid Payload` code described above. Additionally, the comment field could contain information about which files were missing.

7 Sections

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

7.1 Bilingual

The Bilingual section contains bilingual resources, such as XLIFF 2 files. Task types that use this section should further specify what restrictions exist on its resources.

Resources in the Bilingual section are contained in a payload directory named `bilingual`.

7.2 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.

Resources in the Input section are contained in a payload directory named `input`.

7.3 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.

Resources in the Output section are contained in a payload directory named `output`.

7.4 Structured Translation Specification

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

Resources in the Structured Translation Specification section are contained in a payload directory named `sts`.

7.5 TM

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

Resources in the TM section are contained in a payload directory named `tm`.

If the Translation Candidates module has been declared in the manifest, XLIFF 2 TM data in the Translation Candidates module will take precedence over data in this section.

7.6 Terminology

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

Resources in the Terminology section are contained in a payload directory named `terminology.`

7.7 Reference

The 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.

Resources in the Reference section are contained in a payload directory named `reference`.

The Reference section contains a specialized form of File resource called `ReferenceFile`. A `ReferenceFile` contains all the same information as a regular File, but also contains an optional `languageChoice` attribute to indicate if the reference data is specific to the source or target language.

7.8 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 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

Resources in the Preview section are contained in a payload directory named `preview`.

7.9 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.

Resources in the Metrics section are contained in a payload directory named `metrics`.

7.10 Extras

The Extras section contains resources that are not appropriate for any other section. Use of the Extras section is reserved for custom task types. Task types that use this section should further specify what restrictions exist on its resources.

Resources in the Extras section are contained in a payload directory named `extras`.

8 Standard Task Types

This section enumerates the TIPP Tasks currently defined as part of the Linport project effort.

8.1 Translate-Strict-Bitext

The Translate-Strict-Bitext task type represents the translation of bilingual content in the form of XLIFF 2 files.

8.1.1 Task ID

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

http://schema.interoperability-now.org/tipp/v2.0/tasks/translate-strict-bitext

8.1.2 Modules

TIPP packages with the Translate-Strict-Bitext task type allow a TIPP to list XLIFF 2 modules in the manifest. The identifier for the modules to use in the manifest is the module fragment identification prefix according to the XLIFF 2 specification.

8.1.3 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
TM	Optional	Optional
Metrics	Optional	Optional
Reference	Optional	Optional

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

8.1.3.1 Bilingual

The bilingual section must contain one or more XLIFF 2 document instances, which must be XLIFF 2 files containing the content to be translated.

The source and target locales for the XLIFF 2 must be the same the source and target locales in the TIPP manifest.

In the response TIPP, the bilingual section should contain the XLIFF 2 files from the request TIPP, modified with whatever changes were made during translation.

8.1.3.2 Structured Translation Specification

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

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

8.1.3.3 TM

This optional section is used by a request TIPP package 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 either XLIFF 2 (preferred) or TMX 1.4b format. This data is intended to supplement the preferred matches which are carried as part of the Translation Candidates Module of XLIFF 2 file itself. Each XLIFF 2 or TMX file is expected to contain translation units (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.

If data in a Translation Candidates module exists as part of the XLIFF 2 files of the bilingual section, this data will take precedence over data in the TM section.

8.1.3.4 Terminology

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

In addition terminology can be part of XLIFF 2 files of the bilingual section as described in the XLIFF 2 glossary module.

How these two different locations of terminology data within a TIPP package relate to each other (if both exist) is defined in the manifest.

8.1.3.5 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.

8.1.3.6 Preview

This optional section is used by a request TIPP package to carry supplemental resources used in the preview process provided by the XLIFF 2 files. If there is a content relation between the preview files and the files in the bilingual section, the order and the naming of files in both sections must match.

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

8.1.3.7 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.

8.1.4 Processing Expectations

A system that generates a Translate-Strict-Bitext request package expects its XLIFF 2 payload to be processed in accordance with the XLIFF 2 specification, and to receive the processed XLIFF 2 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 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 high-volume translation 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. Accordingly, each daily TIPP may be successfully completed, even though the translation itself is incomplete.
>
> The segment state stored within an XLIFF 2 file (please see http://docs.oasis-open.org/xliff/xliff-core/v2.0/os/xliff-core-v2.0-os.html#state) is both a more powerful and a more natural place to track translation progress. In the XLIFF 2 files of TIPP response packages the state attribute for each segment is mandatory (in XLIFF 2 it is optional)..

8.1.4.1 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 2 itself should be used to determine the extent to which translation was completed.

8.1.4.2 Failure Responses

In addition to standard reasons for failure identified in Success and Failure, a “Task Failure” response should also be provided when the XLIFF 2 resource in the payload is missing or invalid. Additional information about the failure may be provided in the Comment field of the manifest.

8.2 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-processable.

8.2.1 Task ID

The Prepare-Specifications task is identified by the URI

http://schema.interoperability-now.org/tipp/v2.0/tasks/prepare-specifications

8.2.2 Supported Package Object Sections

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

Section	**Request TIPP**	**Response TIPP**
Input	Required	Optional
Structured Translation Specification	Required	Required

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

8.2.2.1 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.

8.2.2.2 Structured Translation Specification

In a request TIPP, the *STS* section should contain an STS file in the format defined by ISO 11669 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.

8.2.3 Processing Expectations

As with the Translate-Strict-Bitext task, the “Success” and “Failure” values of a Prepare-Specifications response package do not indicate anything about the extent to which the input resources were modified.

8.2.3.1 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.

8.2.3.2 Failure Responses

In addition to standard reasons for failure identified in Success and Failure, a “Task 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.

9 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 automated quality assurance processes.

9.1 Defining a Custom Task Type

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

A URI that uniquely identifies the custom task type.

The resource sections supported by request and response TIPPs of the custom task type.

For each supported resource section, a description of what resources may be contained in that section. For example, a custom task type using the Bilingual section should indicate whether it can contain any bilingual file format, or only a specific format such as XLIFF 2.

Processing expectations for the task type, including

- The conditions under which a TIPP response may indicate "Success".
- What modifications to the resources, 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 be used as a model.

9.2 Use of the Extras Resource Section

The Extras resource section is reserved solely for use in custom task types. It exists to allow custom tasks to carry resources that do not logically fit within another section.

Any use of the Extras section should be carefully described in the custom task type definition.

10 Interoperability and Compliance

Assessing compliance with TIPP is complicated by the task type mechanism. While all TIPP implementations are required to produce structurally correct TIPPs, it is not a requirement that every implementation implement every standard task type. Furthermore, because TIPP may be used by different types of tools, not every implementation that supports a task type needs to support the creation of both request and response TIPPs of that type.

For example, a machine translation system might accept content for translation by processing TIPPs with the Translate-Strict-Bitext type. The system would accept request TIPPs of that task type and generate response TIPPs, but it might not have a need to ever create request TIPPs of its own, or to process responses to those requests. Therefore, defining compliance in terms of completely implementing all possible roles in a given task would be a mistake.

Instead, this document defines the basic requirements of "TIPP Compliance" in the abstract, and then provides a vocabulary to allow tools to describe their support for individual task types, as well as the interoperability guarantees provided by those task types.

10.1 TIPP Validity and Compliance

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

Read and write valid TIPPs, as prescribed by this document.

Read and write valid TIPP manifests, as prescribed by this document and the `TIPPManifest.xsd` XML schema.

Reject TIPP packages as prescribed in the Success and Failure section of this document that contain invalid manifests or payloads.

10.2 Unsupported Task Types

When receiving a request TIPP with a task type that the implementation does not recognize or does not support, generate a valid response TIPP with the `Unsupported Task Type` response code. The response TIPP should contain the original request TIPP package.

10.3 Requesting Tools and Responding Tools

A tool may implement a given task type as a requesting tool, a responding tool, or both.

- A requesting tool creates request TIPPs of a given task type and processes their responses.
- A responding tool processes request TIPPs of a given task type and creates responses.

Tools that support TIPP should clearly document the task types they support as well as whether they can act as a requesting tool, a responding tool, or both, for each of those task types.

10.4 Task Types Interoperability Guarantees

Additionally, task types themselves can provide different levels of interoperability through the strictness of their requirements.

Evaluating compliance with a particular TIPP task type depends on the task type itself. Task types can provide one of two types of interoperability guarantee:

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 standard 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 Prepare-Specifications task type offers Syntactic Interoperability.

11 Implementation Guide

11.1 Naming convention for files

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

Name	Description
`manifest.xml`	TIPP manifest
`resources`	Package Object Container

11.1.1 Naming Restrictions

In order to minimize platform-specific incompatibilities, the names of the Envelope and all files within the payload are restricted to the following subset of ASCII:

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

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

Hint: The actual names of the files transferred in a TIPP package of course can contain any kind of characters. For compatibility reasons the files are not stored with these names inside the TIPP package, but there actual names can be kept within the name area in the Resource Information area of the TIPP manifest.

11.2 Language Identifiers

The `SourceLanguage` and `TargetLanguage` elements contain the language identifiers for the source and target languages in the TIPP. These identifiers must conform to the xs:language XML schema type, and must additionally conform to the format described by RFC 5646 or its successor.

11.3 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 "Joe's Translation Management System", with version "2.0" and ID "http://joestms.biz". The specific semantics of Tool IDs are left up to the tool makers.

11.4 TIPP Creators

Information about the systems that generate the task and response packages are encoded in TIPP manifests via the `Creator` element. Request TIPPs identify their creator, while response TIPPs identify both their own creator and the creator of the request to which they are responding. A TIPP creator is identified by three pieces of information:

- `Name`: The common name for the endpoint, such as the name of the controlling organization
- `ID:` An ID, expressed as a URI, such as the URI of the specific system that generated the package
- `Date:` A timestamp, recording the time when the package was created. This value is formatted as described in Format of Date/Time Fields.

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

11.5 Format of Date/Time Fields

Several *manifest.xml* fields contain date and time data. All of these fields must conform to the xs:dateTime XML Schema data type, and should be 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 2 and the format of the XLIFF 2 date attribute.

Other date/time formats are treated as errors.

11.6 Format of Resource Paths

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

- The path is relative to the top-level package folder corresponding to the containing `ResourceSection`. The path should not include the name of this folder. For example, for `File` elements within the `Input` section, all `Location` 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 TIPP 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.

12 Normative References

- XLIFF core 2 2013 http://docs.oasis-open.org/xliff/xliff-core/v2.0/csprd01/xliff-core-v2.0-csprd01.html
- ETSI GS LIS 004 V2.0.0 (2012-07): Global Information Management Metrics eXchange Volume (GMX-V) http://www.etsi.org/deliver/etsi_gs/LIS/001_099/004/02.00.00_60/gs_LIS004v020000p.pdf
- RFC 4122; "A Universally Unique IDentifier (UUID) URN Namespace"; http://www.ietf.org/rfc/rfc4122.txt
- "GNU gettext", https://www.gnu.org/software/gettext/manual/index.html
- "Translation parameters" ASTM F2575 (https://www.astm.org)
- "TBX-Basic" a dialect of ISO 30042:2019 (https://www.tbxinfo.net)
- RFC 5646; "Tags for Identifying Languages"; https://tools.ietf.org/html/rfc5646
- "W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes", http://www.w3.org/TR/xmlschema11-2/