iiRDS Plugin for the DITA Open Toolkit

Contents

Introduction

The plugin *org.iirds.dita.package* for the DITA Open Toolkit (DITA-OT) generates iiRDS packages from a DITA map or a single topic.

iiRDS is a standard for the delivery of intelligent information in the scope of user assistance for products. The information is provided with the product for the purpose of assisting the users in setting up, operating, and maintaining the product. Intelligent information is defined as technical documentation content enriched with metadata.

iiRDS consists of:

- A vocabulary for the metadata provided with the content. The vocabulary is specialized for and restricted to the domain of technical documentation or user assistance. iiRDS uses an RDF schema as technical format. RDF stands for Resource Description Framework and is a standard model for data interchange on the Web.
- A package format for the exchange of packages with intelligent information between different systems, for example, web portals or content delivery servers. The package format uses ZIP and has a predefined folder structure for content and metadata in RDF format.

For details about the iiRDS standard, see https://www.iirds.org

To create iiRDS packages from DITA, this plugin uses a predefined mapping of DITA elements and attributes to iiRDS classes and properties, see Metadata extraction on page 6.

The plugin has the following basic characteristics:

- Based on DITA 1.3
- Supports DITA content in XML format.
- Provides the new transformation type `iirds`.
- Extends the HTML5 plugin.
- Parameters of the HTML5 transformation can be used to integrate custom HTML formatting.
- Resulting iiRDS packages conform to version 1.2, unrestricted, of the iiRDS standard.
- iiRDS metadata is generated based on default metadata elements and attributes in DITA.

The plugin uses extension points provided by the DITA-OT to implement its functionality. Custom parameters and new extension points are provided that can be used to further customize the functionality of this plugin.

Note: DITA 2.0 content, including lightweight DITA in other formats than XML, are not yet supported. Transformations of such content may still work, but this is not guaranteed.

Requirements

This plugin was developed and tested with DITA-OT 3.7.3. To guarantee that this plugin works, the following versions are required:

- DITA-OT 3.7.x
- Java 1.8.x or higher

The plugin can be used with a default version of the DITA-OT or a customized DITA-OT version with specialized document type definitions and modified transformation scenarios.

Install the iiRDS plugin offline

The iiRDS plugin can be installed offline from a ZIP archive or online from the DITA-OT registry or the https://iirds.org website. This topic describes the offline installation.

The `org.iirds.dita.package-1.0.0.zip` archive is available on your local computer. The path to the sources is available at https://www.iirds.org.

1. In the `plugins` directory for your DITA-OT installation, create a new directory `org.iirds.dita.package`
2. switch to the created directory and unpack the `org.iirds.dita.package-1.0.0.zip` archive.
3. In the `bin` directory of your DITA-OT installation, open a command-line client.
4. In the command-line client, run `dita install -v`.

The org.iirds.dita.package plugin is installed in your DITA-OT.

You can build iiRDS packages from your DITA content.

Related information

https://www.dita-ot.org/3.7/topics/plugins-installing

Install the iiRDS plugin from the DITA-OT registry

The iiRDS plugin can be installed offline from a ZIP archive or online from the DITA-OT registry. This topic describes the online installation.

Your computer is connected to the internet.

1. In the `bin` directory of your DITA-OT installation, open a command-line client.
2. In the command-line client, run `dita install org.iirds.dita.package`.

The org.iirds.dita.package plugin is installed in your DITA-OT.

You can build iiRDS packages from your DITA content.

Related information

https://www.dita-ot.org/3.7/topics/plugins-installing

Build and check an iiRDS package

You can use the DITA-OT to build an iiRDS package for a DITA map or a single topic. The output directory will contain the generated iiRDS file with the extension `.iirds`.

1. In the `bin` folder of your DITA-OT installation, open a command-line client.
2. In the command-line client, enter the `dita` command to transform your DITA map or topic using the `iirds` transtype.

   Example: `dita --input myContent.ditamap --o out -f iirds`

   You can apply conditional filtering and use additional parameters as needed.

   An iiRDS package with the following file name is created in the output directory: `<file_name_of_ditamap>.iirds`.
3. Verify that the iiRDS package contains the desired result:
   a) Use your ZIP tool to extract the content from the iiRDS file. If necessary, change the file extension to `.zip` first.

b) In the subfolder `META-INF`, open the file `metadata.rdf` and check if the iiRDS metadata contains the expected values from your DITA content.
c) In the subfolder `content`, open the file `index.html` in a browser and check whether the HTML content is as expected.

If the iiRDS package does not contain the expected result, you can try one of the following:

- If content is missing or you have unwanted content, check your conditional processing attributes.
- If the iiRDS metadata is not what you expected, check the metadata mapping topic to find out why. In case you need a different outcome, you need to customize the metadata extraction.
- If your HTML does not look the way you want, try to use the properties of the HTML5 transformation.

Tip: Integrate the configured transformation into your editor of choice.

Tip: Use the iiRDS Validation Tool to check the integrity of the generated iiRDS package.

Related concepts

Metadata extraction on page 6

Related tasks

Customize metadata extraction on page 12
You can add a customized Java library for metadata extraction that implements the `IirdsMetadataHandler` interface.

Customize HTML5 output on page 13
The iiRDS plugin allows you to customize the HTML5 content of the iiRDS package.

Architecture

Once the iiRDS plugin has been installed in the DITA-OT, you can use the plugin to generate iiRDS packages from your DITA map. The processing steps are described in the following figure:

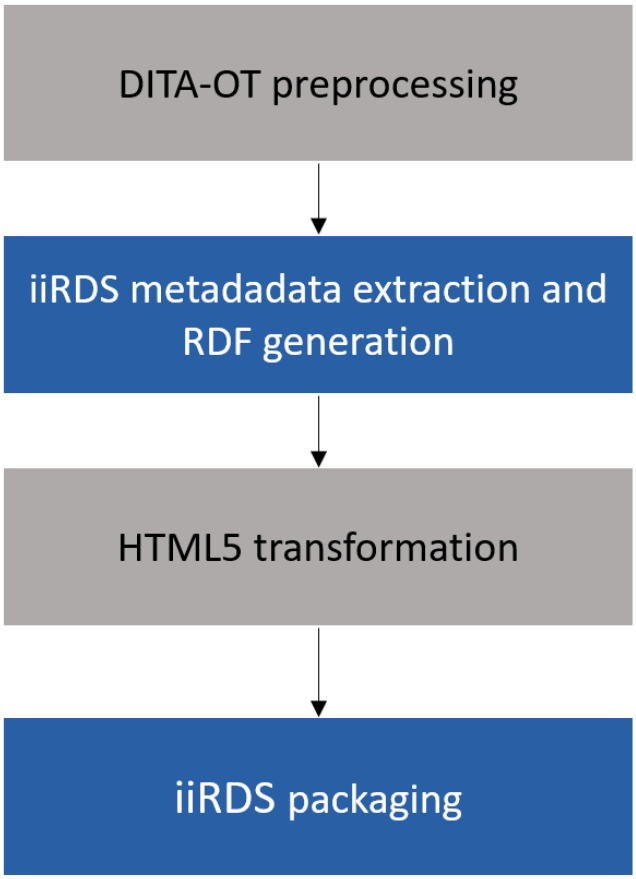

Figure 1: Processing steps of the iiRDS plugin

In detail, this means the following:

- The default preprocessing by the DITA-OT is performed. During the preprocessing, references are resolved, the DITA mechanisms for inheriting metadata from parent nodes are applied and the DITA content is filtered, if required. For details, see https://www.dita-ot.org/dev/reference/preprocessing.
- iiRDS metadata is extracted from the preprocessed XML and the RDF file for the iiRDS package is generated. For details, see Metadata extraction on page 6.
- The default HTML5 transformation is applied to the preprocessed XML. The parameters of this transformation can be used to customize the generated HTML. For details, see Customize HTML5 output on page 13.
- The HTML content and the RDF file are packed in an iiRDS container file that can be imported in any application that supports iiRDS 1.2.

Metadata extraction

The iiRDS plugin comes with a number of predefined metadata handlers to extract metadata from DITA and map them to iiRDS metadata. The iiRDS plugin can process DITA maps or individual topics.

In the RDF file with the iiRDS metadata, the plugin creates the following RDF resources:

- An `iirds:Package` and an `iirds:Document` for the root DITA map, if available. The `iirds:DocumentType` is set to `iirds:OperatingInstruction`.
- An `iirds:Topic` for each topic in a DITA map or for a single DITA topic. For each topic, an additional `iirds:InformationObject` is generated.

In addition to the information units, the plugin creates a directory of the structure of the DITA map, where each entry in the `index.html` of the resulting HTML output is represented by an `iirds:DirectoryNode`.

By default, iiRDS metadata is extracted from the attributes of specific elements or from the text of specific elements. See Metadata mapping on page 7 for a detailed description of the mapping from DITA content to iiRDS and Behavior of the default metadata handler on page 9 to understand how the metadata is mapped.

Metadata mapping

The following table describes how DITA elements and attributes are mapped to iiRDS resources in the RDF file of the iiRDS package. If nothing else is stated, available metadata is mapped to `iiRDS:Document` and `iiRDS:Topic` in the RDF file. The mapping also applies to any specializations of the listed elements and attributes, unless a specific mapping is provided for the specialized type.

Table 1: Mapping of DITA to iiRDS metadata

DITA	iiRDS RDF	Comment
`<ditamap>`	`iirds:Document` with document type `iirds:OperatingInstructions`	All DITA maps are treated as operating instructions because DITA does not provide a document type by default.
`<topic>`, `<task>`, `<concept>`, `<reference>`	`iirds:Topic` with topic type set to `iirds:GenericConcept`, `iirds:GenericTask`, `iirds:GenericConcept`, `iirds:GenericReference`	Topic types are derived from the `@class` attribute of the corresponding topic. Specialized topics fall back to the most basic class.
`<shortdesc>`	`iirds:Topic > iirds:has-abstract`	Content of topic-level short description is used as abstract property. If `<shortdesc>` is wrapped in `<abstract>` and multiple `<shortdesc>` elements are present, only the first short description is evaluated. **Note:** Short descriptions on map level are not evaluated. The DITA standard defines that short descriptions should be copied from topic references to the corresponding topics during preprocessing, but currently this is not the case with DITA-OT 3.7.4.
`<title>`	`iirds:title`	`<title>` elements of root map and topic roots are used. Other `<title>` elements are not evaluated, for example, section or table titles.
`@xml:lang`	`iirds:language`	Language attribute of root map or topic root is used.

DITA	iiRDS RDF	Comment
<prodname> or @product	iirds:ProductVariant	All applicable values of <prodname> elements and @product attributes from the root map and topics are used.
@audience or (<audience> with @type and/or @experiencelevel)	iirds:Role and iirds:Skilllevel	@audience attribute of root map or topic root is mapped to iiRDS:Role. For <audience>, the content of @type is mapped to iiRDS:Role and @experiencelevel is mapped to iiRDS:SkillLevel.
<component>	iirds:Component	<component> elements of root map and topic root are used.
<copyright> with @year and @holder	iirds:rights	<copyright> elements of root map and topic root are used.
<created> with @date	iirds:dateOfCreation	@date attribute of created element of root map or topic root is used.
<revised> with @modified	iirds:dateOfLastModification and iirds:dateOfStatus	@modified attribute of created element of root map or topic root is used.
<created> or revised with @golive	iirds:datefOfEffect	@golive attribute of created or revised element of root map or topic root is used. If both <created> and <revised> with @golive are present, then the latest <revised> value is used.
<created> or <revised> with @expiry	iirds:datefOfExpiry	@expiry attribute of created or revised element of root map or topic root is used. If both <created> and <revised> with @expiry are present, then the latest <revised> value is used.
<topichead>	iirds:DirectoryNode	<topichead> elements are title-only entries in a navigation map. Therefore, they do not have an iirds:Topic equivalent, but they are included as a directory node. The @navtitle attribute or <navtitle> element becomes the label of the directory node.

DITA	iiRDS RDF	Comment
`<navtitle>`	`iirds:DirectoryNode`	`<navtitle>` elements from `<topicref>` elements, `<topichead>` elements, or from a topic are used as labels for the corresponding iiRDS directory nodes.

Related concepts

Unique identifiers on page 11

Behavior of the default metadata handler

If an iiRDS package generated from DITA content does not contain the expected metadata, note the following about the behavior of the metadata handler in the iiRDS plugin:

- The metadata handler extracts the metadata that is present in the preprocessed XML. The DITA standard defines which metadata is propagated from maps to topics or gets inherited from parent topics. The DITA standard also defines whether multiple values for the same metadata element or attribute are merged into a single value or whether multiple values can be aggregated in a topic. To change this behavior, implement a custom metadata handler.
- The metadata handler processes attribute values as they come. If you use a subjectscheme map to control attribute values and this subjectscheme map contains navigation titles with verbose labels for the attribute values, then the resulting RDF will only contain the attribute value. Example: A subject scheme defines the value "product-a" for the @product attribute. "product-a" corresponds to the product Name "Awesome Product". The RDF will contain an instance of `iiRDS:Product = "product-a"`. To change this behavior, implement a custom metadata handler.
- Metadata values are often language-dependent. Which product name are usually the same in any language, metadata such as component names or user roles differ from language to language. The plugin has no way of knowing the component "engine" (EN) is the same as the component "Motor" (DE). Therefore, the metadata entries for this component differ in the English and the German iiRDS package even if they mean the same object in the real world. If you need to have the same iiRDS metadata values across languages, you either need to use controlled metadata values in the same language or implement a custom metadata handler that performs an additional processing step during which the values are mapped and harmonized.

To find out how a custom metadata handler can be used with the iiRDS plugin, see Customize metadata extraction on page 12.

Plugin parameters

The following command-line parameters are provided by the iiRDS plugin:

--iirds.contentpath

Defines the name of the folder in the iiRDS package containing all content files. When not set, `content` is used.

Example: `--iirds.contentpath=files`

--iirds.metadatahandler

Defines iiRDS metadata handlers for metadata extraction from DITA. When not set, all known iiRDS metadata handlers are applied. The parameter value is a + separated list of iiRDS metadata handler names. When a list is provided, then only the listed metadata handlers are used. Priority of the handlers is defined by the order of the list from left to right.

The following example extracts metadata only for `shortdes` and `prodname` DITA elements: `--iirds.metadatahandler=shortdesc+prodname`.

The following metadata handlers are valid default list values. The default list values can be combined with custom metadata handlers.:

shortdesc	Sets `iirds:has-abstract` to the value of the DITA `shortdesc` element.
prodname	Sets `iirds:relates-to-product-variant` to the value of the DITA `prodname` element.
component	Sets `iirds:relates-to-component` to the value of the DITA `component` element.
product-p	Sets `iirds:relates-to-product-variant` to the value of the DITA `product` attribute on the root element.
critdates	Sets `iirds:dateOfLastModification`, `iirds:dateOfCreation`, `iirds:dateOfEffect`, and `iirds:dateOfExpiry` to the value of the DITA child elements of the `critdates` element.
copyright	Sets `iirds:rights` to the value of the DITA `copyright` element.
audience	Sets `iirds:iirds:relates-to-qualification` to the value of the DITA `audience` element.
audience-p	Sets `iirds:iirds:relates-to-qualification` to the value of the DITA `audience` attribute on the root element.
default	Applies all metadata handlers. Allows to combine custom metadata handlers with the default handlers.

--iirds.irihandler

Defines which IRI handlers to be used for assigning IRIs to metadata, information units and information objects. The parameter value is a + separated list of IRI handler names. Priority of the handlers is defined by the order of the list from left to right.

The following example combines a custom IRI handler with the default IRI handler: `--iirds.irihandler=customirihandler+default`.

Related tasks

Customize metadata extraction on page 12
You can add a customized Java library for metadata extraction that implements the `IirdsMetadataHandler` interface.

Customize IRI generation on page 13
You can add a customized Java library for IRI generation that implements the `IRIHandler` interface.

Unique identifiers

To exchange metadata between different systems and map content from one source or language to content from a different source or language, stable and unique identifiers are required. In iiRDS, topics with different metadata or content must have different unique identifiers.

iiRDS metadata is provided as RDF, which uses Internationalized Resource Identifiers (IRIs), the internationalized form of Uniform Resource Identifiers (URIs). DITA does not use IRIs so that unique and stable IRIs have to be derived from the DITA content. Because topic IDs are not unique in all contexts, they cannot be used as IRIs on their own. For example, the same topic may have a different content after conditional filtering has been applied. Therefore, iiRDS topics that are generated from the same source but with different conditional filters, require different IRIs.

The following mechanisms apply:

- The IRI of each `iirds:Topic` is consists of the `@id` value of the topic and a hash of the content. The hash is generated after preprocessing and before rendering. The hash is stable so that the same value is generated with each call of the iiRDS transformation, provided that the topic content has not changed.
- The IRI of the `iirds:Document` consists of the `@id` value of the DITA map and a hash of the content. The hash is generated after preprocessing and before rendering. The hash is stable so that the same value is generated any with each call of the iiRDS transformation, provided that the map content has not changed.

  If a DITA map has no `@id` value, then a UUID is generated and used as the IRI of the `iirds:Document`. This UUID is not identical with each call of the iiRDS transformation.
- For each `iirds:InformationUnit`, an `iirds:InformationObject` is created. This `iirds:InformationObject` has an IRI, which is derived from the topic ID. All information units with the same `@id` value in DITA have the same `iirds:is-version-of` property that refers to this information object.
- The IRI of the `iirds:Package` is a generated UUID.

In addition to this default behavior, the iiRDS plugin provides an extension point that allows to implement other mechanisms for assigning IRIs, for example, to include an additional lookup from other data sources or specialized attributes.

Related concepts

IRI and metadata handlers on page 12

Customization

The iiRDS plugin can be customized to your specific requirements. The plugin provides the following ways of customization:

- DITA-OT Ant extension points
- Java interface for metadata extraction and IRI generation
- HTML5 DITA-OT build parameters

DITA-OT Ant extension points

The iiRDS plugin provides the following extension points that use Ant targets:

`iirds.extractMetadata.pre`	Runs an Ant target after the DITA-OT preprocessing and before the iiRDS metadata extraction.
`iirds.extractMetadata.post`	Runs an Ant target after the iiRDS metadata extraction. Can be used to access the `metadata.rdf` file.
`iirds.package.pre`	Runs an Ant target after the HTML was built and before the iiRDS package is generated.

For details on adding ANT targets to extension points, see https://www.dita-ot.org/3.7/topics/plugin-antpreprocess.

Java interfaces

The iiRDS plugin provides the following Java interfaces:

`IirdsMetadataHandler`	Allows to implement project-specific metadata extraction for all RDF resources from DITA content.
`IRIHandler`	Allows to implement project-specific IRI generation for all RDF resources.

For details, see Customize metadata extraction on page 12 and Customize IRI generation on page 13.

HTML5 build parameters

The iiRDS plugin extends the HTML5 transtype and supports the DITA-OT HTML5 parameters. For example, you can add project-specific CSS, inject XML into the HTML5 header element, or use custom XSLT.

For an example, see Customize HTML5 output on page 13.

IRI and metadata handlers

The iiRDS plugin supports multiple ways to extract metadata out of DITA content and generate IRIs for RDF resources. Metadata is extracted by the plugin's metadata handlers. IRIs are generated by the plugin's IRI handlers.

When the DITA files are processed, the iiRDS plugin tries to extract metadata and generate an IRI based on a sequence of metadata and IRI handlers. The sequence of handlers defines the priority of the handlers. When building iiRDS packages, the `iirds.metadatahandler` and `iirds.irihandler` parameters specify the sequence of handlers. When a parameter is not set, the default fallback is used.

When a handler does not provide an IRI, then the next handler in the sequence is called. As a default fallback, the iiRDS plugin generates unique UUIDs as IRIs for topics, documents, metadata, and information objects.

Custom handlers can be built by extending the Java interfaces `IirdsMetadataHandler` and `IRIHandler`.

Note:

When a metadata handler extracts metadata for a topic which can only be assigned once and another custom metadata handler in the sequence extracts conflicting metadata, then the custom metadata handler has to resolve the conflict.

Related concepts

Unique identifiers on page 11

Related reference

Plugin parameters on page 9

Customize metadata extraction

You can add a customized Java library for metadata extraction that implements the `IirdsMetadataHandler` interface.

- You have a Java library that implements the Service Provider Interface `org.iirds.dita.ot.plugin.spi.IirdsMetadataHandler` and `org.iirds.dita.ot.plugin.spi.IirdsMetadataHandlerProvider`. See https://docs.oracle.com/javase/8/docs/technotes/guides/jar/jar.html#Service_Provider.
- Your Java library is integrated into the DITA Open Toolkit by extending the `dita.conductor.lib.import` extension point. See https://www.dita-ot.org/3.7/topics/plugin-javalib.
- Your content contains metadata for your metadata handler to process.

A project wants to build an iiRDS package using a project-specific metadata extraction to anticipate specialized DITA elements.

1. In the `bin` folder of your DITA-OT installation, open a command-line client.
2. In the command-line client, enter the `dita` command to transform your DITA map or topic using the `iirds` transtype and the `iirds.metadatahandler` parameter to use your metadata handler.

   Example: `dita --input myContent.ditamap --o out -f iirds --iirds.metadatahandler=YourMetadataHandler+default`

An iiRDS package with the following file name is created in the output directory: `<file_name_of_ditamap>.iirds`. The iiRDS package contains metadata based on the project-specific requirements.

Customize IRI generation

You can add a customized Java library for IRI generation that implements the `IRIHandler` interface.

- You have a Java library that implements the Service Provider Interface `org.iirds.dita.ot.plugin.spi.IRIHandler` and implements and registers `org.iirds.dita.ot.plugin.spi.IRIHandlerProvider`. See https://docs.oracle.com/javase/8/docs/technotes/guides/jar/jar.html#Service_Provider.
- Your Java library is integrated into the DITA Open Toolkit by extending the `dita.conductor.lib.import` extension point. See https://www.dita-ot.org/3.7/topics/plugin-javalib.
- Your provide IRI mappings if required by your IRI handler.

A project wants to build an iiRDS package using a project-specific IRI mapping. The IRI mapping is provides as a CSV file. The CSV file is stored next to the DITA map.

1. In the `bin` folder of your DITA-OT installation, open a command-line client.
2. In the command-line client, enter the `dita` command to transform your DITA map or topic using the `iirds` transtype and the `iirds.irihandler` parameter to use your IRI handler.

   Example: `dita --input myContent.ditamap --o out -f iirds --iirds.irihandler=YourIriCsvHandler+default`

An iiRDS package with the following file name is created in the output directory: `<file_name_of_ditamap>.iirds`. The iiRDS package contains IRIs based on the project-specific requirements.

Customize HTML5 output

The iiRDS plugin allows you to customize the HTML5 content of the iiRDS package.

- Custom XSLT is available.
- Custom CSS is available.

A project wants to generate an iiRDS package that contains custom HTML5 content that uses the project's CSS.

1. In the `bin` folder of your DITA-OT installation, open a command-line client.
2. In the command-line client, enter the `dita` command to transform your DITA map or topic using the `iirds` transtype and the HTML5 parameters to use your CSS and XSLT.

   Example: `dita --input myContent.ditamap --o out -f iirds --args.cssroot=absolutPathToMyCssFolder --args.css=myCustom.css --args.copycss=yes --args.csspath=outputCssFolderName --args.xsl=relativePathToMyCustomStylesheet.xsl`

An iiRDS package with the following file name is created in the output directory: `<file_name_of_ditamap>.iirds`. The iiRDS package contains HTML files generated by the project's XSLT and using the project's CSS.

License and contact information

The plugin is licensed under the open-source license „Apache License, Version 2.0“, including the accompanying documentation and its source code. For details, see https://www.apache.org/licenses/LICENSE-2.0.

The plugin was developed for the iiRDS consortium by Empolis Information Management GmbH, testing and documentation was done by parson AG.

In case of questions, contact the iiRDS consortium, see https://www.iirds.org/contact.