Trifolia Workbench

Table of contents

Introduction

Trifolia is an open source tool for creating FHIR profiles and CDA templates. Users can export templates/profiles to:

- MS Word (DOCX)
- Web (HTML)
- Excel (XLSX)
- Other XML formats

Trifolia exposes FHIR DSTU1, DSTU2, and STU3 REST APIs, as well as a native API.

Trifolia is available to download and install. Users should know Trifolia requires the following technologies:

- Microsoft C# .NET 4.5
- OAuth 2.0 Authentication
- NET MVC and Web API
- SQL Server 2012+
- JS and Angular.JS
- Bootstrap

The public code repository is available at https://github.com/lantanagroup/trifolia.

What's New

Version 5.3.8

Released 10/19/2020

Markdown

Trifolia has migrated to using a rich-text syntax called Markdown, instead of Markup & XHTML. In fields such as a template's description, a constraint's narrative text, or volume 1 section text, a Markdown editor is used in place of the previous Markup & XHTML editors. Markdown is a more common syntax that is widely used, and is used by FHIR standards.

Volume 1 content in MS Word exports

The Microsoft Word exports now include the option to include Volume 1 content (which can be defined by editing the Implementation Guide). This is a pre-release of the functionality, as there may still be issues to work out that we have not yet discovered; particularly with regards to the formatting and styling of the content within the MS Word document that gets exported.

Multi-tasking template edits

For those that like to open multiple "template edit" windows at the same time, we have made it easier for you to identify which template is in each tab/window by appending the template's title to the title of the tab.

Development Log - 5.3.8

Type	Summary
Defect	The new markdown description field doesn't let you see all your text and no v
Defect	Markdown tables in Word export not formatting properly
Defect	Selecting multiple parent templates for an export, exports full IG instead of just th

Improvement	Schematron MAY validation

Development Log - 5.3.7

Type	Summary
Improvement	Validation of identifier (templateId) extension (yyyymmdd)
Improvement	Update Trifolia CDA Schema with latest SDTC and Form Definition extensions

Development Log - 5.3.6

Type	Summary
Defect	Importing value sets from VSAC creates new value sets with the wrong identifier ty

Development Log - 5.3.5

Type	Summary
Defect	Navigation bar hides fields
Defect	Can't export a CDA IG - APTA Companion Guide for C-CDA R2.1
New Feature	Generate document template overview table in MS Word document

Development Log - 5.3.4

Released 01/12/2019

Type	Summary
Defect	Users should not be allowed to close the popup dialog during value set import
Defect	Incorrect UMLS licensing error when attempting to export
Defect	VSAC import doesn't work with codes that have a long display name
Defect	Constraints marked as "header" with a template reference is not exported to MS

Development Log - 5.3.3

Released 11/07/2018

Type	Summary
Defect	Autogenerated samples of attributes where xsi:type has been constrained do not include the
Defect	Using the toggle fullscreen button in the template editor's markdown text box breaks scrolling
Defect	Value set "Source" is not set when importing from PHIN VADS
Defect	VSAC import doesn't work with codes that have a long display name

Development Log - 5.3.2

Released 09/14/2018

Type	Summary
Defect	Importing from PHIN VADS results in error when multiple versions has the same effectiveTime
Improvement	Update HQMF R2 schema to latest from Sept 2017

Development Log - 5.3.1

Released 08/09/2018

Type	Summary
Defect	Value set tables not producing an anchor - links to the value set not working
Defect	Update styling on markdown so that special words would be styled appropriately for the MSW
Defect	Captions in Exported Word documents are improperly including a colon and space
Defect	Move the anchor/bookmark for tables to wrap the “title” of the table

Development Log - 5.3.0

Released 07/25/2018

Type	Summary
Defect	Copying/moving templates to an IG with a different IG Type does not update the template's "t
Defect	Disabled template editor still allows you to edit value set binding date
Defect	Update CDA schema to latest version
Defect	Export: Templates/Profiles: Select all check box
Defect	Trifolia allows there to be One oid for two different code systems
Defect	Removing an import value set generating error
Defect	Terminology Search saying "no value set found" when value set exists
Defect	Vocab: When values in a value set are retired, they don't export with an IG that was published
Defect	Constraints Overview Table - repeat header across pages
Defect	Value Sets in this Guide table - Reduce width of Code column
Improvement	Migrate from WIKI Markup & XHTML to Markdown
Improvement	Restrict Template/Profile binding to appropriate types
New Feature	Prominently show release information in pop-up window
New Feature	Append template's name to browser's title bar
Task	Refactor the logic for determining the base url of a FHIR version

System Requirements

- Chrome
- Safari
- Firefox
- Internet Explorer 9+

Language

CDA Term	FHIR Term

Template	Profile
Branch	Slice
Branch Identifier	Discriminator

Logging In

Open-Source Authentication

The existing Active Directory and custom HL7 authentication has been replaced with a generic authentication method (OAuth 2.0). **OAuth** is an open standard for authorization, commonly used as a way for Internet users to log in to third party websites using their Google, Facebook, Microsoft, GitHub, etc. accounts without exposing their password.

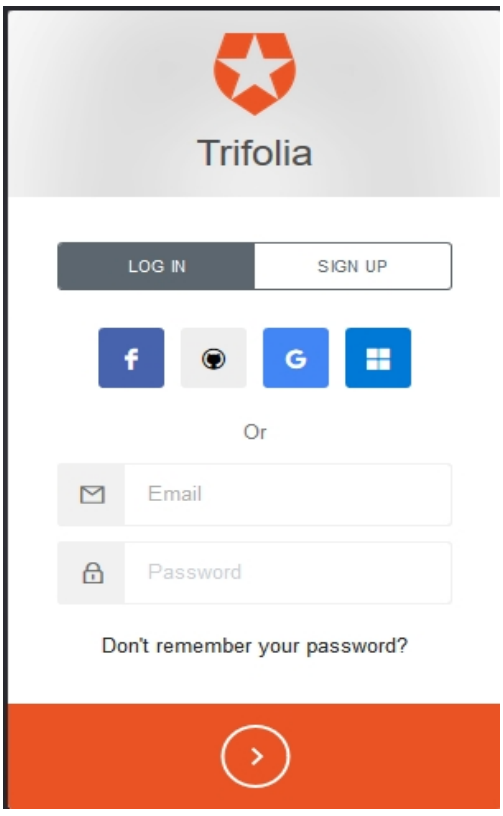
As part of this migration, all existing Trifolia user accounts have been added/imported to the new OAuth Trifolia directory. Existing Trifolia users will receive a Password Reset link via e-mail that will allow them to use their Trifolia credentials when logging in using the Auth0. Clicking on the link in the email will direct the user to auth0.com and prompt the user to create a new password for your Trifolia account.

Existing users can use their Trifolia credentials to login, and pre-existing permissions to resources (implementation guides, templates, value sets, code systems, etc.), will persist. Each authentication method (Facebook, Microsoft Account, Username/Password, etc.) represents a separate user in Trifolia, with a separate username/password and permissions.

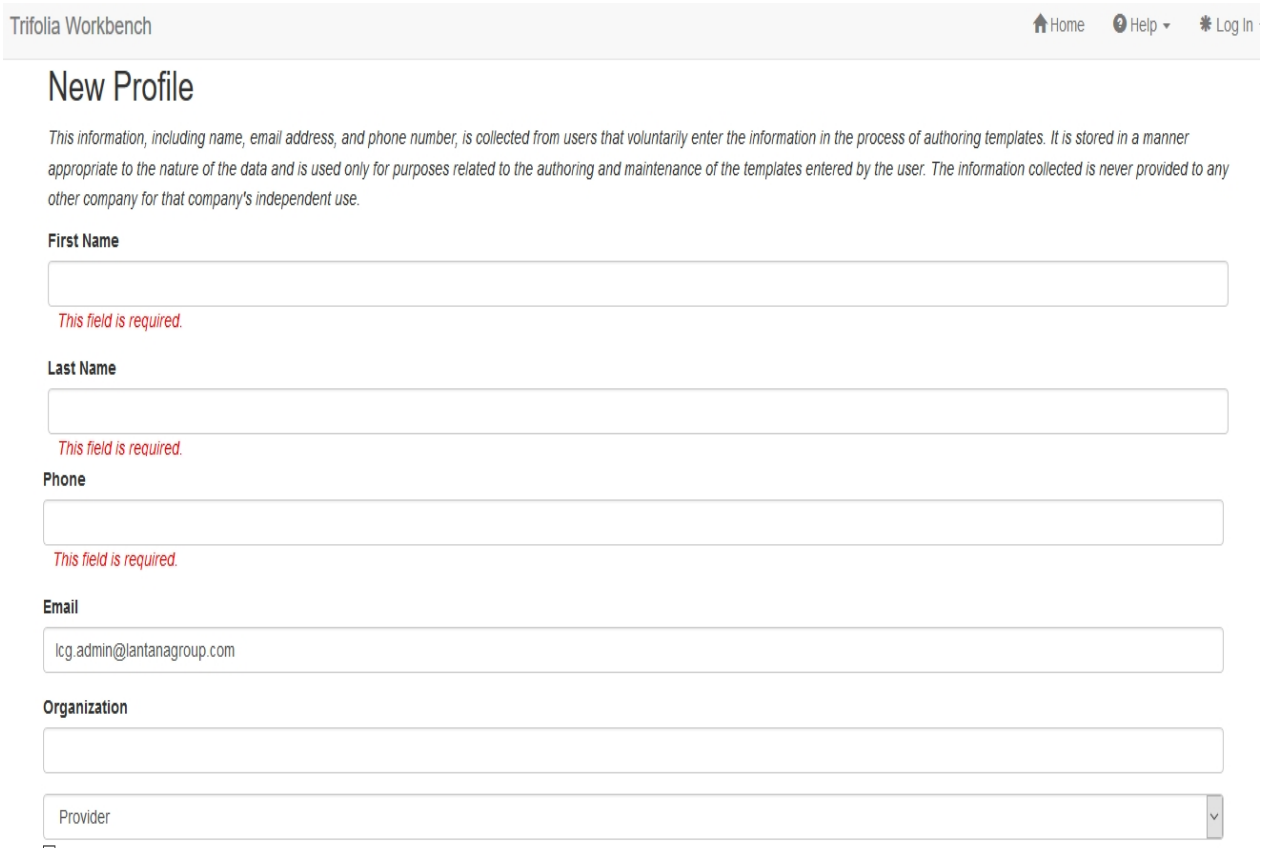
Please refer to the Security section for more information on access and permissions.

Log In to Trifolia

1. In the upper-right corner, select Log In and choose Login from the drop-down menu. When users click the "Login" option. The Auth0 Log In page appears, as shown below



2. Once you log in to Trifolia Workbench, the functions available depend on the access permissions granted to your login credentials.
3. If it is the first time logging in, you will be prompted to enter "My Profile" information. See My Profile for more information. If are an existing user,
you can modify your My Profile information from the menu in the top-right corner.



Navigation

Use the Trifolia Workbench menus to access all available features. Note that editors and administrators have additional features that are not available to Read-only accounts. Menus are arranged across top of the window, in tabbed fashion.

- Home - The home page provides a brief overview of Trifolia Workbench and informs users what features are new in the current version.
- Browse - Browse and edit implementation guides, templates/profiles, value sets, and code systems. See The Browse Menu.
- Export - Export data in a number of formats. See The Export Menu.
- Import - Allows users to select an XML file for import. This XML file should be the "Proprietary" export format of an implementation guide and its associated templates. The "Import" permission is set with the "Administrators", "Template Authors" and "IG Admins" roles by default.
- Reports - Choose from a variety of report types, then generate, format, and export or print a report. See The Reports Menu.
- Administration - Perform administrative tasks. See The Administration Menu.

- Help - Provides the help topic for the current page. If you are on the home page, you can view the complete help system with a navigation tree.

Terminology

Below is some information about the value sets and code systems used in Trifolia exports.

Value Set Members

Trifolia treats value sets as though they are fully expanded. When Trifolia presents a value set that contains another value set, it displays all the members of the value set in an expanded view. This represents a fully expanded view of the value set.

Value Set Status & Date

Each member of a value set contains status and date fields. Date and Status fields are used to determine which members will be exported to XML and MS Word output.

Calculated Date

For each member of a value set, a date is calculated, and this date, along with the status determines whether to include that member in an export. The date is derived using this priority:

1. If the constraint includes a value set binding date, it is used.
2. If no value set binding date is present, the implementation guide's publish date is used.
3. If no implementation guide publish date is available, the XML or MS Word export date is used.

Status

The value set member status can be entered when a new value set member is added. Once the date is calculated, the status (if entered) and date are used to determine whether the member is included in the export as follows:

- The member is included in the export if the status is Unspecified (empty) or the calculated date is prior to the date for the member's active status.
- The member is excluded from the export the calculated date falls on or after the member's Inactive status date.

Example

Given the following value set:

Code	Display Name	Code System	Status	Date
Value 1	Display 1	SNOMED	Unspecified (blank)	Unspecified
Value 2	Display 2	LOINC	Active	1/1/2013
Value 3	Display 3	SNOMED	Active	1/1/2013
Value 2	Display 2	LOINC	Inactive	3/1/2013

For an export generated on 1/15/2013, the resulting list is:

- Value 1
- Value 2
- Value 3

For an export generated on 5/1/2013, the resulting list is:

- Value 1

- Value 3

Code Systems and Value Sets

Trifolia creates a relationship between a code system and a value set member. Code systems are used to reference a name/identifier combination.

A code system is referenced in one of the following ways:

- A value set's member is a member of a specified code system
- A constraint indicates that an element/attribute in a template/profile must be selected from a code system
- A constraint, in combination with the constraint value, indicates that the value is a member of a specified code system

Vocabulary XML

Vocabulary XML is used for Schematron validation. Value sets are bound to constraints in the Template Editor. Only a value set with a Binding value of **static** is validated by the Schematron. When you export vocabulary XML, only a value set with a Binding value of **static** is included. A value set with a Binding value of **unspecified** (empty) or **dynamic** is ignored. For more information on binding value sets see Bindings in the Constraints section.

User Profiles

My Profile

The "My Profile" screen shows the data associated with your currently logged-in user. When a user logs into Trifolia for the first time, the My Profile screen appears, and the user completes the required fields. This information is specific to Trifolia. The information captured in the user profile is used in several ways:

- To display a user's name as an author on templates/profiles
- On the "Edit Implementation Guide" screen's permissions
- By Lantana to determine who can be contacted with information relating to Trifolia news/announcements/inquiries

The following fields are captured on the "My Profile" screen (* = required):

- First Name*
- Last Name*
- Phone*
- Email*
- Organization
- Organization Type
- "It is OK to contact me" - This indicates if it is OK for Lantana to email or call the user regarding Trifolia news/announcements/inquiries.

Changes to the "My Profile" screen are not persisted until the "Save" button is selected.

VSAC/UMLS Credentials

VSAC requires that you have a license to UMLS to import value sets from VSAC and to export implementation guides that contain VSAC content.

To register for a VSAC/UMLS license, click here. The registration process requires approval from NLM. This typically takes two days to complete.

After your UMLS/VSAC license request is approved, you can enter your credentials for VSAC/UMLS in Trifolia's My Profile screen. Use the "Test" button to confirm that you have entered the correct UMLS/VSAC credentials, prior to saving changes in "My Profile".

Trifolia encrypts and stores your VSAC/UMLS username and password so that you do not have to repeatedly enter your UMLS/VSAC credentials every time you need to import from the VSAC and export an implementation guide that contains VSAC content. Your VSAC/UMLS credentials are *never* used outside of the context of Trifolia and are not shared with third parties.

Security

Trifolia uses two different methods of security:

- **Role-based security** - Controls what a user can do. This includes all actions and controls. It controls what a user can select to initiate an action. Roles are assigned on a system-wide basis, meaning that the role does not change for the different areas of functionality within the application.
- **Permissions-based security** - Controls what a user can view or edit. This controls the user's ability to view and edit implementation guides and Templates/Profiles. Permissions are assigned to implementation guides using the Permissions tab in the Implementation Guide Editor. See the Permissions section in Authoring Implementation Guides.

Organizations

Trifolia organizes users and groups for each configured organization. For example, Lantana Consulting Group is configured to be an organization that has access to Trifolia, using it's own authentication method, and granting access to implementation guides and templates/profiles to users within that organization. HL7 is another organization configured to have access to Trifolia with its own set of users and roles.

Roles

Administrators can Assign users to roles. Roles determine which functionality (or securables) the user can access. Roles are specific to a particular organization.

The following roles are always available.

- Administrators - Access to all securables.
- Template/Profile Authors - Access to viewing, editing and exporting of templates/profiles, code systems, and value sets, and viewing of a number of reports.
- Users - View and export implementation guides, value sets, and code systems.
- IG Admins - View, edit, and export implementation guides, and view and export vocabulary and Schematron.

Administrators can also create additional roles, selecting sets of securables that are specific to those roles.

Permissions

Overview

Permissions can be specific to an implementation guide. Two kinds of permissions are available:

- **View Permissions** - Grant permission to a user or group to view a particular implementation guide.
- **Edit Permissions** - Grant permission for a user or group to edit the templates/profiles in a particular implementation guide.

Implementation guides can control who has permissions to edit templates owned by the implementation guide.

- User can indicate that a user can have view and/or edit permissions
- User can indicate that a group can have view and/or edit permissions
- User can indicate that an entire organization can have view and/or edit permissions
- User can indicate that a user or group from another organization can collaborate on templates within the IG.

Requesting Permissions to Access an Implementation Guide

To view and/or edit an IG, you must be granted permission to access. Permission is granted by the Editor of the IG. Only those IGs to which you have access will display in the list of available IGs. IG Editors may allow users to “request access”. The list of IGs shown to the user in the "Request Access" screen includes only IGs that have their “can request access” option enabled. If you have not been granted access, you will see a link below the search field on the Browse Implementation Guides screen to "request access".

Fields in the "Request Access" Window

- Access Level: Indicates if you would like "view" or "edit" permissions to the IG.
- Message: An optional message that is sent to the access manager of the IG.

Request Access to an IG

- Select thee access level you would like to have for the IG
- Optionally, specify a message to send to the access manager.
- Click the "Request" button.

After submitting the request, an email is generated and sent to the access manager of the implementation guide. The email message will include your name and email address, to indicate to the access manager *who* is requesting access, along with the permission requested and the (optional) custom message.

The message sent to the access manager follows this format:

```
User <FirstName> <LastName> (<Email>) has requested access to
<ImplementationGuide> on <Date>.
The user has requested <Permission> permissions.

Message from user: <Message>

Use this link to grant them permission to the implementation guide: <Link>
Use this link to deny them permission to the implementation guide: <Link>
```

Note: The request alone does not guarantee that you will be granted permission to an IG; it is the decision of the access manager to grant permissions.

Access Managers and Requests

If an IG is configured to allow access requests, an Access Manager should be specified. An Access Manager is responsible for handling requests from other users to access an IG. Requests to access an IG are

1. sent as an email notification to the Access Manager, using the email address associated with the Access Manager's user account
2. available in the Trifolia web application, after a user is logged in, by selecting the user's menu in the top-right of Trifolia and selecting "Access Requests".

Viewing/Approving/Denying Access Requests

When a request is made, the email notification sent to the Access Manager includes a link to approve and a link to deny the access request. Upon clicking one of these links

1. Trifolia is opened in a browser
2. Authentication is initiated (if not already logged in)

3. Permissions are either granted or denied (depending on which link the Access Manager selected)
4. The user that made the request is notified via email that their permissions have been either granted or denied

In addition to an email notification, the requests can be viewed/approved/denied by selecting the "Access Requests" menu item under the user's menu in the top-right corner of Trifolia.

Note: If no access requests are pending your approval, and you do not have any outstanding requests yourself, the "Access Requests" menu item is not available.

Upon selecting "Access Requests", a pop-up window is displayed with (up to) two separate tabs. The first tab represents access requests that are pending *your* approval, the second tab represents access requests you have made that are pending *another* external user's approval.

The "Pending Approvals" tab provides two buttons to "Approve" and "Deny" access requests. Upon selecting one of these actions, the user is (if approved) granted the specified permissions to the IG and emailed a notification, or (if denied) emailed a notification indicating their access request has been denied.

Groups

Each user can create their own groups, and can request to join any existing groups. These groups are used by implementation guide permissions to assign multiple users read/edit access to an implementation guide and its templates/profiles.

To manage groups, select the menu for your name in the top-right corner of Trifolia, and click "My Groups".

The top portion of the screen shows groups that you manage as well as groups that you are joined to.

The bottom portion of the screen shows groups that you may request to join.

If you are a manager of a group, you will have options to Edit and/or Delete the group. If you are only a member of the group, you will have options to view the group, but no options to edit the group.

Editing a group

Create a new group by clicking the "Add" button in the top-right of the "My Groups" screen.

Edit an existing group (if you are a manager of the group) by clicking the "Edit" button to the right of the group.

Name	The name of the group that is displayed to all users
Description	The description is shows to all users so that they know what the group represents
Anyone can join	When checked, anyone can join the group without approval, and joining the group is automatic as soon as the user clicks "Join". When not checked, an email is sent to the managers of the group to indicate which user would like to join the group. The managers are responsible for logging into Trifolia and adding the user to the group.
Disclaimer	Disclaimers are shown in two places: 1. When users request to join the group, the disclaimer is shown to the user prior to them joining. When the user clicks "Join", they are effectively agreeing to the terms of the disclaimer. 2. Users that are members of a group with a disclaimer are shown the group's disclaimer under the "Help" > "Disclaimers" menu.
Managers	The list of users that are responsible for managing the group. These users can edit the details of the group, including adding/removing other managers and members.
Members	The list of members that belong to the group. These members are granted permissions to implementation guides when the group is assigned permissions to the implementation guide.

Tutorials

This section includes general tips, tricks and tutorials on how to use some of Trifolia's more advanced features.

Building a FHIR IG

This tutorial will provide an overview of the workflow/process used to design/develop a FHIR implementation guide using Trifolia.

The basic steps are as follows, with details below:

1. FHIR and IG Publisher Overview
2. Create an implementation guide.
3. Create profiles for the implementation guide.
4. Upload "FHIR resource instances" to the implementation guide.
5. Export the implementation guide as a FHIR build package.

FHIR and IG Publisher Overview

See the FHIR current build for more information.

FHIR (*Fast Health Interoperability Resources*) is designed to enable information exchange to support the provision of healthcare in a wide variety of settings. The specification builds on and adapts modern, widely used RESTful practices to enable the provision of integrated healthcare across a wide range of teams and organizations.

- Implementation guides are a collection of FHIR resources(profiles, value sets, examples and human readable documentation) tied together by the Implementation Guide resource
- Acronym FHIR = F – Fast (to design & to implement) H – Health, I – Interoperable and R – Resources(building blocks)
- Resources are discrete chunks of clinical information, can be assembled into larger constructs and resources are operated on via FHIR's REST APIs
- FHIR documents are a collection of resources bound together, and are transferred between systems as a Bundle resource
- The root of a FHIR document is a Composition resource that can be signed, authenticated, etc, and has the same basic obligations as a CDA document.

IG Publisher

See the IG Publisher Documentation for more information.

- The FHIR team provides an IG Publishing tool that takes the implementation guide resources and converts them to a set of 3 different types of files (generated resources in xml, json, ttl formats, a set of fragments ready to include in generated html files and Several different zip files: which are used by implementers for various purposes).
- The outcome of the publishing process is a set of HTML files that represent the implementation guide. These files can be posted to a web server.

- Alternatively, you may use the IG publisher to validate and render a set of Profiles, value sets etc without building a formal IG

Create an implementation guide

See Authoring > Implementation Guides for details.

- The "type" of implementation guide is set to a FHIR implementation guide type.
- Create a base identifier for the implementation guide that is a URL (ex: https://trifolia.lantanagroup.com/my_base_id"). This base url will be used by the FHIR IG publisher, and requires that all profiles within the implementation guide have a matching base identifier.
- Provide a description for the implementation guide. This description (if specified in the Implementation Guide Editor) is on the front page of the implementation guide after the IG publisher has been run on the artifacts produced by Trifolia.
- Include additional guidance that will not be inferred by the profiles of the implementation guide in the "Volume 1" tab, by adding sections to "Volume 1". Each of the sections defined in the implementation guide's "Volume 1" tab will be displayed on the "Overview" tab of the implementation guide's front-page.

Create FHIR profiles for the implementation guide

See Authoring > Templates/Profiles > Editor for details.

- Each FHIR profile must have the same base url as the implementation guide in the "long id" field. The identifier for FHIR profiles is defaulted to a combination of the bookmark/name of the FHIR profile and the associated implementation guide's base url.
- Each profile must have a "short id" that matches the outer-most leaf level of the long id (everything to the right of the last / in the long id).
- Define constraints for each profile in the "Constraints" tab of the editor, modifying the cardinality, value set bindings, etc. as appropriate.

Defining and using extensions

In the event that an extension is necessary, you must produce a profile that describes how the extension is intended to be used, and reference that extension as a "Contained template/profile" on the "extension" element of another profile.

Upload "FHIR resource instances" to the implementation guide

See Authoring > ImplementationGuides > Files for details.

In certain cases, (such as "Conformance", "SearchParameter" and "Questionnaire") instances of resources need to be uploaded to Trifolia, then associated with the implementation guide rather than defining profiles. Add these "FHIR resource instances" to the implementation guide's files (as type "FHIR Resource Instance") so that they are included in the FHIR build package when exported from Trifolia.

Export the implementation guide as a FHIR build package

See Exporting > XML/JSON for details.

Export the implementation guide as a FHIR build package. The zip package that will be run against the FHIR ig publisher.

Run the FHIR IG publisher by following these steps:

1. Extract the FHIR IG Package to a directory on your machine.
2. Download the FHIR IG Publisher jar file from the FHIR's Downloads page or click here to directly download the .jar file. Store the .jar file in the same directory that the FHIR IG Package was extracted to.
3. Execute the "RunIGPublisher.bat" batch script.
4. If using the GUI interface, select the XXXX.json file exported from the ZIP.
5. Press "execute".

Note: The FHIR IG publisher is new functionality, being actively developed. If the latest version of the FHIR IG publisher produces errors, you should seek guidance from http://chat.fhir.org in the #Implementers channel.

FHIR Extensions

This tutorial describes how to create and use an extension in FHIR.

1. Create a new template/profile for the extension
    1. Select the “Extension” for the “Applies To” field
    2. Create a constraint for the “@url” attribute with a single-value binding of the same URL as the template/profile’s “Long ID” (ex: “http://hl7.org/fhir/us/ccda/StructureDefinition/CCDA-on-FHIR-Authorization”)
    3. Create a constraint for the “value[x]” choice. On this constraint, I suggest selecting “SHALL” and “1..1” for the conformance and cardinality, because an optional value[x] would not provide much value in an extension, and may not even be valid according to FHIR
    4. Create a constraint for *one* of the “value[x]” choice options (ex: “valueString”)
2. In the template/profile where you want to use the extension:
    1. Create a constraint for the “extension” element
    2. Select the extension template/profile created in step 1 (above) as a contained template
    3. Select that the “extension” constraint is a branch/slice “root” if this template/profile might have multiple extensions. Otherwise, this will be the only extension that can be used in the template/profile.

Template Relationships

Consider the following example:

Implementation Guide A

Template A

Template B

Implementation Guide B

Template C

Template D

References Template A

Template A will show up in both Implementation Guide A and B, because Template D references (either contains, or implies) template A . If you remove the reference to template A from Template D, then template A will no longer show up in Implementation Guide B.

Inferred Templates

Inferred templates are templates not directly owned by a given implementation guide, but are referenced by it.

Example Scenario

Implementation Guide A directly contains **Template A, B** and **C**

Implementation Guide B directly contains **Template D, E** and **F**

Template D contains **Template A** from **Implementation Guide A**

For "Implementation Guide B", "Template A" is an "inferred template" because it is not directly owned by "Implementation Guide B" but it is referenced by a template/profile that is directly owned by "Implementation Guide B".

Design Patterns

This page is used to describe patterns that may be re-used in template/profile design. The goal of this page is to demonstrate methods of writing computable constraints for more complicated scenarios.

Alternative Entry Templates

Scenario: You have a section that needs to contain one of two entry templates, or both.

Example

1. SHALL contain 1..* entry
    1. SHOULD contain 1..1 entry, such that it
2. SHALL contain 1..1 XXXX (template: oid)
    1. SHOULD contain 1..1 entry, such that it
3. SHALL contain 1..1 YYYY (template: oid)

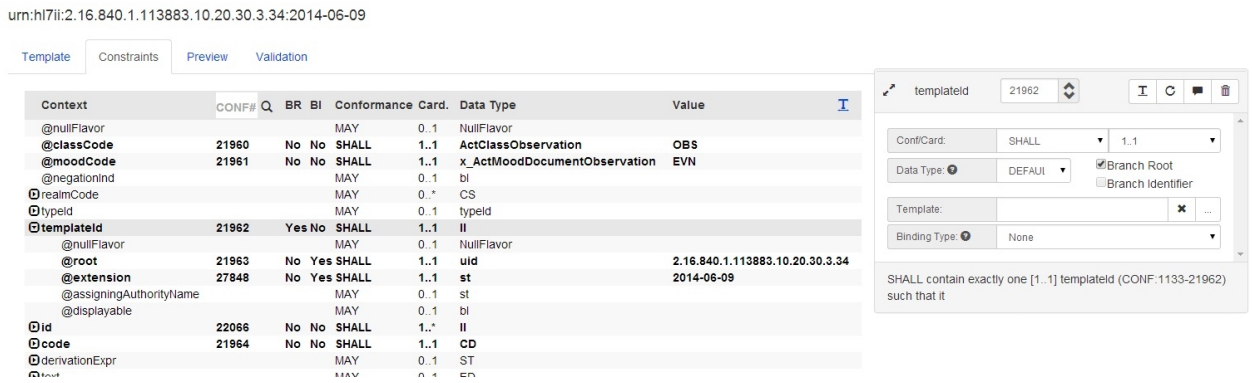
Explanation

- The first constraint requires that at least one entry be present. It does not care what type of entry, just that at least one exists.
- The second and third constraints indicate what types of entries you should use.

CDA Best Practices

Assert the Template Identifier as a Constraint(s)

1. On every template, create a constraint for "templateId"
    1. Create a constraint for "@root" equal to the OID of the template's identifier
    2. Create a constraint for "@extension" equal to the extension of the template's identifier (if any)



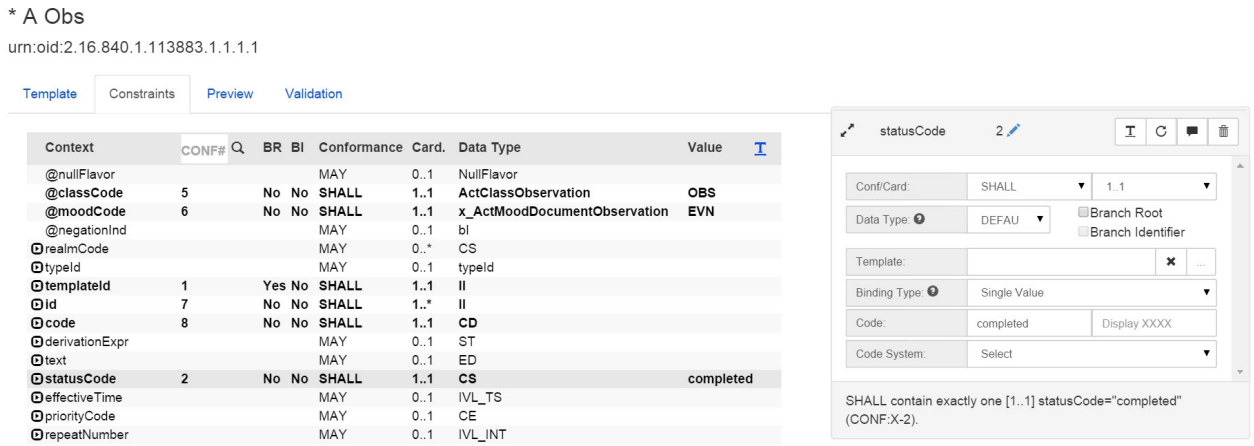
Context	CONF#	BR	BI	Conformance	Card.	Data Type	Value
@nullFlavor				MAY	0..1	NullFlavor	
@classCode	**21960**	**No**	**No**	**SHALL**	**1..1**	**ActClassObservation**	**OBS**
@moodCode	**21961**	**No**	**No**	**SHALL**	**1..1**	**x_ActMoodDocumentObservation**	**EVN**
@negationInd				MAY	0..1	bl	
realmCode				MAY	0..*	CS	
typeId				MAY	0..1	typeId	
templateId	**21962**	**Yes**	**No**	**SHALL**	**1..1**	**II**	
@nullFlavor				MAY	0..1	NullFlavor	
@root	**21963**	**No**	**Yes**	**SHALL**	**1..1**	**uid**	**2.16.840.1.113883.10.20.30.3.34**
@extension	**27848**	**No**	**Yes**	**SHALL**	**1..1**	**st**	**2014-06-09**
@assigningAuthorityName				MAY	0..1	st	
@displayable				MAY	0..1	bl	
id	**22066**	**No**	**No**	**SHALL**	**1..***	**II**	
code	**21964**	**No**	**No**	**SHALL**	**1..1**	**CD**	
derivationExpr				MAY	0..1	ST	

Constraints with Data-type CS

Coded Simple Value (CS)

Definition: Coded data in its simplest form, where only the code is not predetermined. The code system and code system version are fixed by the context in which the CS value occurs. CS is used for coded attributes that have a single HL7-defined value set.

This means that you should only specify the "Code" attribute for a single-value binding. An example of where this is commonly the case is with the "statusCode" element.



Context	CONF#	BR	BI	Conformance	Card.	Data Type	Value
@nullFlavor				MAY	0..1	NullFlavor	
@classCode	**5**	**No**	**No**	**SHALL**	**1..1**	**ActClassObservation**	**OBS**
@moodCode	**6**	**No**	**No**	**SHALL**	**1..1**	**x_ActMoodDocumentObservation**	**EVN**
@negationInd				MAY	0..1	bl	
realmCode				MAY	0..*	CS	
typeId				MAY	0..1	typeId	
templateId	**1**	**Yes**	**No**	**SHALL**	**1..1**	**II**	
id	**7**	**No**	**No**	**SHALL**	**1..***	**II**	
code	**8**	**No**	**No**	**SHALL**	**1..1**	**CD**	
derivationExpr				MAY	0..1	ST	
text				MAY	0..1	ED	
statusCode	**2**	**No**	**No**	**SHALL**	**1..1**	**CS**	**completed**
effectiveTime				MAY	0..1	IVL_TS	
priorityCode				MAY	0..1	CE	
repeatNumber				MAY	0..1	IVL_INT	

Constraints on nullFlavor

Flavors of null are defined as follows:

NI No information. This is the most general and default null flavor.

NA Not applicable. Known to have no proper value (e.g., last menstrual period for a male).

UNK Unknown. A proper value is applicable, but is not known.

ASKU Asked, but not known. Information was sought, but not found (e.g., the patient was asked but did not know).

NAV Temporarily unavailable. The information is not available, but is expected to be available later.

NASK Not asked. The patient was not asked.

MSK There is information on this item available but it has not been provided by the sender due to security, privacy, or other reasons. There may be an alternate mechanism for gaining access to this information.

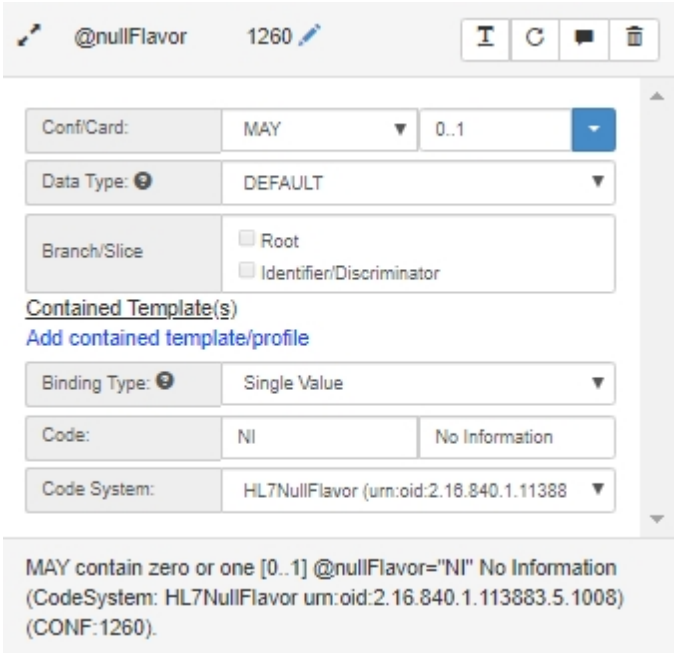
OTH The actual value is not an element in the value domain of a variable. (e.g., concept not provided by required code system).

Some use cases may justify optionally allowing for a nullFlavor [0..1], and requiring that the appropriate value for nullFlavor attribute be present (i.e. bind to single-value binding) when nullFlavor is present. In order to achieve this:

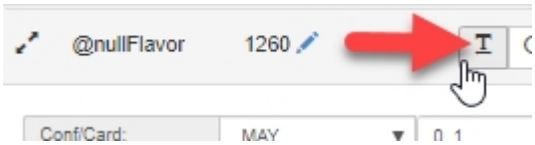
1. Create constraint on nullFlavor with MAY Cardinality [0..1]

2. Select the binding type of 'Single Value Binding'

3. Enter the appropriate Code, Display, and Code System for the flavor of null that will be required when nullFlavor is present



4. Add a Child Primitive constraint to nullFlavor by selecting the underscored 'T' icon in the constraint editor (right)



5. Select you're newly created primitive constraint. Add 'If nullFlavor is present, nullFlavor SHALL be 'NI'' into the text box.

6. In the bottom right 'View Mode', change view to Engineer, de-select 'Auto Generate' and enter to the custom schematron logic in order to technically validate that the desired value of nullFlavor is found when nullFlavor is present.

For the example above, the constraint states "MAY contain zero or one [0..1] @nullFlavor="NI" No Information (CodeSystem: HL7NullFlavor urn:oid:2.16.1.113883.5.1008)". The custom

schematron to validate that nullFlavor has a value of 'NI' in all cases when nullFlavor exists is: "test=not(nullFlavor) or nullFlavor='NI'"

FHIR Best Practices

Bindings on Data-type Elements

When binding terminology (single-value, value set or code system) to an element in a FHIR profile, it is preferred that the binding be applied to the data-type elements. Trifolia gives you the freedom to assign the binding to elements/attributes *within* data-types, but this is not preferred.

Bad Example #1

1. SHALL contain 1..1 name
    1. SHALL contain 1..* coding
    2. SHALL contain 1..1 code, which SHOULD be selected from ValueSet AdvanceDirectiveTypeCode urn:oid:2.16.840.1.113883.1.11.20.2 STATIC 2006-10-17

Bad Example #2

1. SHALL contain 1..1 name
2. SHALL contain 1..* coding, which SHOULD be selected from ValueSet AdvanceDirectiveTypeCode urn:oid:2.16.840.1.113883.1.11.20.2 STATIC 2006-10-17

Good Example

1. SHALL contain 1..1 name, which SHOULD be selected from ValueSet AdvanceDirectiveTypeCode urn:oid:2.16.840.1.113883.1.11.20.2 STATIC 2006-10-1

Formatting Text

Trifolia allows you to format descriptions, notes, and narrative constraint text using Markdown syntax. When you render the output to HTML or Word, the text is rendered according to the specified export format.

Fields That Support Wiki Syntax

The following fields support wiki syntax formatting:

- Templates
    - Description
    - Notes
- Constraints
    - Primitive Constraint Narrative
    - Constraint Description
    - Constraint Heading Description
- Implementation Guides
    - Volume 1 Sections
    - Template Type Description (ex: “Document Templates” description)

HTML

The Markdown fields in Trifolia may have a mix of Markdown and HTML. For example:

```
This **is an** example of <span class="font-family: Courier New">mixed html</span> and ma
```

Formatting XML elements

You may want to highlight XML element names so that they show in a different format than other text. A CSS class has been added to accommodate this, which is also respected by MS Word documents.

The CSS class name is "XMLName", and may be used like so:

```
These pilot instance identifiers begin with <span class="XMLName">2.16.840.1.113883.3.117
```

Syntax Guide

Emphasis

```
**bold**
```

```
*italics*
```

```
~~strikethrough~~
```

Headers

```
# Big header
## Medium header
### Small header
#### Tiny header
##### Even more tiny header
###### Super small header
```

Lists

```
* Unordered list item
* Unordered list item
* Unordered list item
```

```
1. Numbered list item
2. Numbered list item
3. Numbered list item
```

Links

```
[alt text](http://example.com)
```

Quotes

```
> This is a quote
> It can span multiple lines!
```

Images

```
![alt text](http://example.com/image.jpg)
```

Tables

```
| Column 1 | Column 2 | Column 3 |
| -------- | -------- | -------- |
| John     | Doe      | Male     |
| Mary     | Smith    | Female   |
```

Embedded Code

```
<ClinicalDocument>
  <setId root="..." />
  ...
</ClinicalDocument>
```

Primitive Constraints

Schematron files are used to validate the content/values of codes and OIDs within an IG by compiling a series of xml rule assertions. Trifolia produces an exportable Schematron file for each IG based on template constraints defined by the Trifolia user. While Trifolia can generate/export Schematron for basic IG constraints, more complex IG architecture may require that custom prose and/or Schematron rules be created in addition to Trifolia's automatically generated prose/Schematron. Every primitive requires custom Schematron to be defined by the user because Trifolia does not have the computable information about the constraint to automatically generate the Schematron for it. Computable constraints *may* also have custom Schematron, but it is not usually necessary.

SHALL primitive within a MAY primitive

An issue within Trifolia to be aware of is for the case where a MAY primitive on a template that contains a SHALL primitive.

Schematron is expected to enforce the SHALL primitive in every instance where the MAY constraint is present. In other words, when MAY primitive is found to be present, the Schematron is expected to subsequently enforce that the SHALL primitive does not break the rule as specified.

The SHALL primitive is enforced (as expected) in cases when the MAY primitive constraint is placed on *a template:*

1. ***MAY*** *contain zero or one [0..1] Certifier performer* `(identifier: urn:hl7ii:XXXX:YYYY)` `(CONF:XXX-XXXX)`*such that it*

a. **SHALL** contain exactly one [1..1] **@root** (CONF:XXX-XXXY).

In this example, Schematron enforces that @root exists when Certifier performer is present; these are both computable constraints that don't have custom prose *or* custom Schematron.

1. ***MAY*** *contain zero or one [0..1] Certifier performer* `(identifier: urn:hl7ii:XXXX:YYYY)` (`CONF:XXX-XXXX`) *when the document type is XXXX, such that it*

a. **SHALL** contain exactly one [1..1] **@root** (CONF:XXX-XXXY).

In this example, "when the document type is XXXX" must be represented by a primitive constraint. Because the parent constraint (the MAY constraint) is a primitive constraint, the Schematron cannot enforce @root exists when Ceritifier performer is present.

Solution 1

1. Create custom Schematron rule at the IG level
2. Suggested: Add a note on the MAY primitive constraint saying that the IG Schematron will check the SHALL primitive is honored when the MAY primitive is present.

Solution 2

Do not nest SHALL primitives within a MAY primitive. Rather, such SHALL primitives should exist as their own constraint at the top level within the template.

Example:

10. The performer containing the Certifier Performer template **SHALL** be present for a live birth certificate but

SHALL NOT be present for a fetal death report (CONF:XXX-XXXXX).

Implementation Guides

Use the Browse Implementation Guides page to see a list of implementation guides and to search for a specific guide or set of guides.

You may request access to implementation guides that are not available to you. See Permissions > Requesting Permissions to Access an Implementation Guide for more details.

Locate an Implementation Guide for Review or Editing

1. From the Browse menu, choose **Implementation Guides**. The Implementation Guides page appears.
2. Use the Trifolia's Automatic Lookup feature to find the implementation guide you want to view or edit. A list of implementation guides that match your search criteria appears.
3. Once you have found the implementation guide you want, select **View**, or **Edit** (depending on your permissions) from the Actions menu for the selected guide. Refer to the Viewing an Implementation Guide details below or Authoring Implementation Guides section for more information.

Viewing an Implementation Guide

In the Browse Implementation Guides page, in the row containing the guide you want to view, select View or WebView in the last column. The selected guide appears in the viewer.

Select a tab to view information about the guide:

- **Templates/Profiles** - View a list of templates/profiles in the guide, with a description of each. Select a template/profile's View or Edit link to view or edit the selected template/profile. Clear the Show Descriptions check box to view only the template/profile names and OIDs.
- **Notes** - View any notes attached to templates/profiles. Use the filters to limit the notes you are viewing.
- **Primitives** - View any primitive text attached to templates/profiles. Use the filters to limit the primitives you are viewing.
- **Audit Trail** - View the audit trail for any changes that have been made to templates/profiles in your guide. Use the filters to limit the audit entries you are viewing.
- **Files** - View a list of files attached to the implementation guide. Download an individual file or download all files at one time.

Templates/Profiles

Use the Templates/Profiles page to view a list of CDA, HQMF and E-Measure templates and FHIR profiles. Use the View Template/Profile page to view the template/profile details.

Viewing a Template/Profile

There are two ways to view a template/profile:

- From the Implementation Guides list
- From the Templates/Profiles list

View a Template/Profile from the Implementation Guides List

1. From the Browse menu, choose **Implementation Guides**.
2. Use the Automatic Lookup feature to find the implementation guide containing the template/profile you want to view. A list of implementation guides that match your search criteria appears.
3. In the implementation guides list, select **View** for the implementation guide you want to view.The

implementation guide appears, showing you a list of templates/profiles contained in the guide.

4. Review the list of templates for the type of template you want to view. Locate the template/profile you want to view and select **View**. The template/profile viewer opens, containing the template/profile details.

View a Template/Profile from the Templates List

1. From the Browse menu, choose **Templates/Profiles**.
2. Use the Trifolia's Automatic Lookup feature to find the templates/profiles you want to browse. A list of templates/profiles that match your search criteria appears.
3. In the Template/Profile list, choose **View** for the desired template/profile. The template/profile viewer opens, containing the template/profile details.

Viewing the Template/Profile

- **Constraints** - View a list of the template/profile's constraints as they appear in the implementation guide.
- **Samples** - View a code sample attached to the template/profile. The "Samples" tab does not appear in the template/profile viewer if no samples exist for the template/profile.
- **Relationships** - View the relationship of the current template/profile to other templates/profiles.
    - **Implied By** the current template/profile - lists any templates/profiles that reference the current template/profile
    - **Contains** a template/profile - lists any templates/profiles contained within constraints in the current template/profile
    - **Contained By** a template/profile - lists any templates/profiles that reference the current template/profile
    - **Implies** a template/profile - lists a single template/profile referenced in the **Implies** field of the current template/profile
- **Changes** - View changes to the template/profile. View the changes **Inline** or as a **List**. (See below)

Viewing Changes - List

When you view changes to a template/profile as a List, each constraint change is listed in a table. The type of change is listed as:

- Added
- Modified
- Removed

Notes:

- Each entry contains the constraint reference number, the old narrative, and the new narrative.

Terminology

Use the Terminology Browser to view and edit existing value sets and code systems, and to create your own.

Browse Available Value Sets and Code Systems

1. From the Browse menu, choose **Terminology**. The Terminology Browser appears.
2. Select the **Value Sets** tab to browse and edit value sets, and the **Code Systems** tab to browse and edit code systems.
3. Use the Terminology Browser's **Search** field to find the value sets or code systems you want to view. A list of value sets or code systems matching your search criteria appears.

4. Select the drop-down menu on the value set or code system to view and edit.

Viewing a Value Set

1. Browse to the value set you would like to view via Browse > Terminology > Value Sets.
2. Select the drop-down menu on the value set you would like to view.
3. Select the "View" menu.
4. The "View Value Set" screen appears, showing the meta-data about the value set, where the value set is used, and the codes/concepts included in the value set.

Concepts

The Concepts tab shows each of the codes/concepts included in the value set.

Only 20 codes/concepts are shown at a time. The paging options at the bottom of the Concepts tab appears when the value set has more than 20 codes/concepts and allows you to page through the codes/concepts in the value set one at a time, or jump to the beginning and end of the list of codes/concepts.

A search option is available where you can enter a keyword and select the "Search" button. The codes/concepts list will refresh to filter only codes/concepts that contain the specified keyword.

Relationships

The Relationships tab displays an expandable table of which implementation guides and templates/profiles use the value set. By expanding a row, you can see what constraints have a binding to the value set.

Authoring

Implementation Guides

Fields for an Implementation Guide

- **Name**: Required. A name for the implementation guide.
- **Display Name**: The display name is used in publishable artifacts. If no display name is specified, the "name" is used instead.
- **Type**: Required. The type of templates/profile that will be stored in the implementation guide (ex: "CDA", "E-Measure" or "FHIR"). The type selected here directly corresponds to the XML schema that is used to build templates/profiles in the template/profile editor.
- **Organization**: Optional. The organization for which the implementation guide is being created
- **Identifier/Base URL:** Required.
- **Consolidated Format**: Exports MS Word documents using the consolidated formatting guidelines. This option is made available to support legacy implementation guides that were published prior to the guidelines defined by the consolidation project.
- **Access Manager**: A user with "edit" permissions that is responsible for receiving access requests and granting access to the IG.
- **Allow Access Requests**: Select "Yes" to make the implementation guide available in the "request access" window (when an access manager is defined). For more information, see Access Requests in the Permissions section.

Creating a New Implementation Guide

1. From the **Browse** menu, choose **Implementation Guides**. The Implementation Guides page appears, showing a list of available implementation guides.
2. At the right of the column header bar, select **Add**. The Add Implementation Guide page appears, as shown below.
3. In the **Name** box, enter a name for your implementation guide.
4. From the **Type** drop-down menu, choose an implementation guide type.
5. Select the **Organization** the IG is being created for.
6. Enter the Identifier/Base URL for the implementation guide that is a URL (ex: https://trifolia.lantanagroup.com/my_base_id").
7. In the **Display Name** box enter the name that displays in forms and reports.
8. Enter the **Web Display Name** for a the name that displays in the Web Viewer.
9. In the **Use Consolidated Constraint Format** box, choose **Yes** or **No**.
10. Optionally specify the **Access Manager**" of the implementation guide (IG), and if you want the IG to be available for access requests. See Access Requests for more details.
11. Optionally specify whether or not to **Allow access requests** for this implementation guides.
12. Select **Save**. The Implementation Guides page appears, and includes the new guide.

Adding Templates / Profiles to an Implementation Guide

1. From the **Browse** menu, choose **Templates/Profiles**. A list of templates/profiles appears.
2. Select the template/profile you want to add to the implementation guide. Use Trifolia's Automatic Lookup features to find the template/profile you want to add.
3. In the **Template/Profile** browser , select the **Edit** icon. The **Edit Template/Profile** page appears with four tabs – Template/profile (active), Constraints, Preview, Validation.
4. Select an implementation guide from the **Implementation Guide** drop-down, or type in the Implementation Guide box to use Trifolia's Automatic Lookup feature to locate the implementation guide to which you want to add the template/profile.
5. At the bottom of the page, select **Save**. The template/profile is added to the implementation guide.

See Authoring > Templates/Profiles for more information

When you set the status of an implementation guide to Published, the templates associated with the guide are locked so they cannot be edited.

Publish an Implementation Guide

1. From the **Browse** menu, choose **Implementation Guides**. The Implementation Guides page appears, showing a list of available implementation guides.
2. Locate the implementation guide you want to publish.
3. Select **View** to the right of the selected guide. The **View Implementation Guide** page appears, showing the details of the selected guide.
4. From the Quick Buttons **Workflow** menu, choose **Publish**. A **Select Publish Date** box appears.
5. Enter a date in the **Publish Date** box, or select a date in the pop-up calendar and select **OK**.
6. The **Publish Date** is set to the specified date, the **Publish Status** is set to **Published**, and *all associated templates/profiles, value sets, and code systems* for the implementation guide are locked. The guide is published.

Delete an Implementation Guide

1. From the **Browse** menu, choose **Implementation Guides**. The Implementation Guides page appears, showing a list of available implementation guides.

2. Locate the implementation guide you want to delete.
3. From the **Actions** menu for the selected guide, choose **View**. The Implementation Guide viewer appears, showing the templates used in the selected guide.
4. Select **Delete** in the Quick Buttons selection. The **Delete Implementation Guide** page appears.
5. Choose a **Replacement Implementation Guide** from the drop-down menu. Templates in the deleted guide are assigned to this guide after you delete.
   Note: If you do not select a replacement guide, no guide is assigned to the templates/profiles in the selected guide. You will need to assign them manually to a template/profile.
6. Select **Delete**! A confirmation message box appears indicating “Are you absolutely sure you want to delete this implementation guide?”.
7. Click **OK** on the confirmation message if you wish to delete the IG, and the implementation guide will be deleted, and the templates/profiles will be assigned to the specified replacement guide.

Permissions

Permissions for an implementation guide determine who can view and who can edit the implementation guide and the templates that the implementation guide owns.

Add Permissions for an Implementation Guide

1. Go to **Browse** > **Implementation Guides**
2. Select **Edit** on the desired implementation guide
3. Select the **Permissions** tab
4. Select **Add** for the desired permission type (either **View** or **Edit**)
5. Select the group shown, or search for an individual user using the **Search** text and **Search** button and select one (or more) of the search results
6. Select the **OK** button to confirm adding the selected concepts as permissions to the implementation guide

Changes to the permissions of the implementation guide are not persisted until the implementation guide's **Save** button has been selected.

The "Add Permission" dialog allows you to select one of the following types of concepts:

- **Group** - a group that is defined within Trifolia for the selected organization.
- **User** - an individual user that belongs to an organization. Individual users are only displayed after searching the organization within the "Add Permission" dialog.

Notify Users About New Permissions

When permitting a user access to the implementation guide, you may select "Notify new users and groups that they have been granted permissions". This option sends an email to any users with new permissions informing them of the change in permissions.

The email message includes their user information, what permission they have been granted, and a link to the imlementation guide they have been granted permissions to.

Notifications are only sent to individual users that have been granted permissions, and users within groups that have been granted permissions to the implementation guide. When sending notifications to a group, be aware of the users that are included in the group. The notification will be sent to *every* user in the group.

Access Requests

When the implementation guide (IG) has an access manager and allows access requests, the display name (or name, if no display name is specified) of the IG will be available in a list of IG's to which a user does not have access. The Browse Implementation Guides screen prompts the user with a message indicating that

they may not have access to an IG they are looking for. When the user selects the "request access" link, a dialog is presented to the user listing each IG that meets the following criteria:

- The IG allows access requests
- The IG has an access manager defined
- The user does not have access to to the IG

Trifolia does not enumerate users from groups in the "access manager" drop-down. In order for a user to be selected as an "access manager", the user must have a separate "edit" permission (and not just be part of a group that has "edit" access to the IG).

Cardinality

You can specify the way in which cardinality appears in constraints.

Template Types

You can specify the way in which document types appear in implementation guides and other exports.

Custom Schematron

You can specify custom Schematron patterns for use in templates.

Categories

During the export of Trifolia artifacts, users can select which constraints can be exported in the artifact using Categories. The Implementation Guide categories allow the grouping of constraints associated with an implementation guide. The constraint editor allows the selection of the IG categories. During the export of template XML, MSWord and Schematron user can select multiple categories of constraints to export. When a category is selected, constraints will be included in the implementation guide that either have no category specified, or a category matching the selected category on the export settings. If no category is selected on the export screen, all constraints will be exported regardless of the constraint's category. The category selection on the export screens allows for multiple selections. Select multiple categories by holding CTRL while you click on each category

An implementation guide may contain one or more categories. These categories, once defined, may be associated with constraints within templates of the implementation guide. Categories may be used to generate different exports, exporting the constraints associated with the selected category.

Create a Category

1. Go to the **Edit Implementation Guide** screen for the implementation guide you want to modify categories for
2. Select the **Categories** tab
3. Specify the name of the category in the "NEW CATEGORY" text field
    - It is suggested that you choose a *short* name for the category (example: "CAT1")
4. Select the **Add** button
5. **Save** the implementation guide

Delete a Category

1. Go to the **Edit Implementation Guide** screen for the implementation guide you want to modify categories for

2. Select the **Categories** tab
3. Select the **Remove** button next to the category you want to remove
4. **Save** the implementation guide

Associate the Category to Constraints

See the Template/Profile > Editor > Constraints > Categories section.

Bookmarks

Bookmarks provide reliable, consistent and unique links to templates and other implementation guide content. Bookmarks can be generated and edited. Bookmarks are included in the exported implementation guides. One bookmark is automatically created for each template.

Use the Edit Bookmarks page, available from the Quick Buttons (found on the View Implementation Guides page) to review and edit the bookmarks used in the selected implementation guide.

Review and Edit Bookmarks

1. From the **Browse** menu, choose **Implementation Guides**. Select **View** on an implementation guide. The **View Implementation Guide** screen appears.
2. Select the **Edit** drop-down menu
3. Choose **Bookmarks** from the drop-down menu. The **Edit Bookmarks** page appears, showing a list of templates, with the bookmark assigned to each.
4. Select **Edit** for the bookmark you want to change. The title and bookmark entries for the selected bookmark change to editable fields.
5. Edit the title and bookmark, the select **Update**. The title and bookmark entries are updated as edited.
6. Select **Return** to return the **View Implementation Guide** page. If you edited the title, your changes appear in the template/profile list.

Files

You can attach files to an implementation guide. Once attached, those files appear in the Files tab of the Files tab of the View Implementation Guide Page.

Attach a File to an Implementation Guide

1. View the implementation guide you want to attach the file to.
2. From the **View Implementation Guide** page, choose **Files** from the **Edit** menu.
3. The **Manage Implementation Guide Files** page appears.
4. Select **Add File**. The Add File window appears.
5. Select **Choose File** and locate the file you want to attach.
6. Select a file type from the Type menu, enter a **Description**, and enter a **Note** about the file, then select **OK**. The file appears in the file list.
7. Select **Save**. The file is uploaded and attached to the implementation guide.

Note: The Notes field is used to track changes to a file. When you upload a new version of the same file, the file is replaced, but the note remains with the file's history.

Remove a File from Implementation Guide

1. View the implementation guide's Manage Files page containing the file you want to remove.
2. Select Remove File next to the file you want to remove. A confirmation message appears.
3. Select Save. The file is removed from the implementation guide.

Edit File Information

4. View the implementation guide's Manage Files page containing the file you want to update.
5. Select Edit Description next to the the file you want to edit. The Edit Description box appears, as shown below.
6. Make changes as needed and select OK.
7. Select Save. The description is updated.

View a File's History

Select History next to the file's name.

Versioning

You can link a version of an implementation guide to a previous version. This allows you to view changes to the implementation guide and all of its templates.

Note: You can create a new version of an implementation guide only if it has been published. Each time you publish a version, you can create a new version and link it to the previous version.

Create a New Version of an Implementation Guide

1. From the Browse > Implementation Guides page, select New. A blank implementation guide appears in the editor.
2. Enter the name of the previous version of the implementation guide.
3. Note: Do not include a version number - it is added automatically.
4. From the Previous Version drop-down menu, choose the previous version of the implementation guide. Select X to clear the previous version.
5. Select Save. The new version of the implementation guide is saved. It contains all the templates from the previous version, but its Publish Status is set to Draft.
6. Once you have saved the new version, the Previous Version link appears in the View Implementation Guide page.
7. Once you publish the new version, you can reference it as a Previous Version in a new implementation guide.

Note about versions: Version history is saved with a template. When a new version of a template/profile is created, use the Changes tab of the Template Viewer to see the changes in the template/profile since the previous version.

Editor

An asterisk (*) indicates that the template/profile has been modified.

Use the template/profile editor to create new templates and to edit existing templates. The editor contains these tabs:

- **Template/Profile** - view and edit template/profile meta-data, such as the name, identifier, type, and status.
- **Constraints** - use a nested view of all elements and attributes from the base standard, and use the constraint editor to add constraints for this template.
- **Preview** - view the template/profile as it looks in the final export. As you make changes to constraints, the preview is updated.
- **Validation** - view validation messages for the template. Validation messages indicate structural recommendations on constraints (such as primitives and branching). Validation is only updated after

saving the template.

Quick Edit

Select "..." and select a template, then select Go to edit it.

View Mode

Choose a role to change way you view the template/profile editor. Template fields are enabled or disabled and editing constraints show or hide fields appropriate to the specified user's role.

- **Analyst**: Modify meta-data about the template/profile and structural information about constraints within the template/profile (add/remove constraints, conformance verbs, cardinality, etc.).
- **Editor**: Modify description, notes, and add for the template/profile and the description, notes, and heading information for constraints. This role is primarily focused on publication-related information.
- **Engineer**: Modify Schematron fields for constraints.

Save

Save the changes you've made in the template/profile editor. Choose from these options when you save:

- Save and Continue - save the template/profile and stay in the template/profile editor.
- Save and Publish Settings - save the template/profile and view Publish Settings for it.
- Save and List - save the template/profile and view the Browse Templates page.
- Save and View - saves the template/profile view the Template Viewer.

Note: When you leave the template/profile editor without saving, a confirm window appears.

Cancel

- Cancel and List - discards changes made to template/profile and view the Browse Templates page.
- Cancel and View - discards changes made to template/profile and view the View Template page.

Meta-Data

Required Fields

- **Name**
- **Identifier** - A unique identifier for the template. Must be less than 255 characters. See Template/Profile Identifier Formats for more details.
- **Bookmark** - A hyperlink anchor used in the MS Word document formatted export to allow cross-referencing. The bookmark must be unique, and must not contain any special characters (including spaces, with the exception of underscores). The bookmark may not be greater than 40 characters.
- **Implementation Guide** - Choose from a drop-down list of implementation guides to assign the template/profile to a guide. The template/profile may also be referenced by other implementation guides (via contained or implied relationships).
- **Type** - Choose from a list of available types (document, section, entry). Templates/profiles are organized by template type when an implementation guide is exported as an MS Word document. When you select a template type, the Applies To field is set. to a pre-determined default appropriate to that type.
    - See FHIR > Extensions for information on building a FHIR extension profile.
- **Applies To** - Choose the location in the base standard/schema where this template/profile can be used. The default is based on the selected template type, but can be customized by selecting the ellipsis

("...") button to the right.

- **Extensibility** - Choose whether instances of the template/profile can have additional information (open), or must contain only what is defined in the template/profile (closed).

Optional Fields

- **Implied Template** - Choose a template/profile that supplies constraints implied in this template. Only those templates with the same Applies To values are available to select.
- **Extensibility** - Choose whether the template/profile is Open or Closed.
- **Status** - Choose the status of the template. Draft, Published, or Deprecated. Available options depend on the status of the implementation guide the template/profile is associated with.
- **Description** - Enter a narrative description of the template. The text appears in an exported implementation guide. This field supports wiki formatting syntax.
- **Notes** - Notes are available for review only by template/profile Analysts and Editors, and are not included in exports.

Template Identifier Formats

Template identifiers can be in one of the following formats:

- urn:oid:XXXX
  Where XXXX is a valid OID (ex: 2.16.1.2.3.4)
- urn:hl7ii:XXXX:YYYY-MM-DD
  Where XXXX is a valid OID
- http(s)://XXXX
  Where XXXX is a web address representing the template

Constraints

The Constraints tab first shows only the element/attributes from the base standard/schema. Element/nodes (or nodes) that are associated with constraints are bold.

After selecting a node in the tree, the constraint editor window opens on the right side of the screen. The view displayed in the constraint editor depends on whether the node has been constrained. The constraints editor view also varies with the "Role" selected for the template/profile editor as a whole (Analyst, Editor, and Engineer).

Computable constraints are constraints based on the element/attributes within the schema that can be represented using the fields supported by Trifolia (such as Conformance, Cardinality, Contained Template, Value Set, etc.). All computable constraints have a context, such as "@classCode".

Primitive constraints are free-text constraints that cannot be represented using the standard computable fields within Trifolia. Primitive constraints are always shown below computable constraints so that the order of computable constraints can be preserved and accurately reflected in exports. Primitive constraints do not have a context; instead, primitive constraints show "N/A" for the context.

- To create a constraint on a node within the tree, select a node and select the + icon (Create Computable) icon in the header of the constraint editor.
- To create a primitive constraint at the top-level of the tree, select the **T** "Add Top-Level Primitive" icon located in the header of the tree, after the Value column.
- To create a primitive constraint within a computable constraint, select a computable constraint and select the **T** "Add Child Primitive" icon in the header of the constraint editor.
- To remove a constraint and return the node to the default definition of the schema, select the Trash can (Remove) icon in the header of the constraint editor.
- To create a note on a constraint (which is only available to template/profile authors, and is not included

in exports), select the “Edit Note” icon in the header of the constraint editor.

- The constraint editor window can be expanded and collapsed with the "Minimize/maximize" in the upper left corner of the Constraint Editor window.

See FHIR > Extensions for information on using pre-defined/reusable extensions.

Numbers

Each constraint has a unique number associated with it.

- The number generated is unique within the implementation guide that the template/profile is associated with.
- The same number can be used on two different constraints, as long as the constraints are in templates associated with different implementation guides.

Trifolia allows you to edit the constraint number, as long as the number follows the criteria specified above.

In addition to the *unique* constraint number, Trifolia allows constraints to have a *Display Number* which does not have to be unique. The Display Number can be contain letters, in addition to numbers (ex: "CAT1-2523").

Note: When creating a new constraint in the template/profile editor, the number will be displayed as **AUTO** until the template/profile is saved. Once the template/profile is saved, the database generates a unique number for the constraint. When the template/profile is refreshed after saving, the number generated will be shown in the constraint.

Edit a Constraint Number

1. Open the **Template Editor** for the desired template
2. Go to the **Constraints** tab
3. Select a constraint
4. In the Edit Constraint pop-up window, select the pencil/edit icon next to the number of the constraint in the heading of the pop-up window.
5. Modify the **Unique Number** or **Display Number** fields
6. Select **OK**
7. **Save** the template

Cardinality and Conformance

The cardinality and conformance rules for conformance constraints in IGs (and entry into Trifolia) have certain rules that need to be followed. The information below uses the new cardinality syntax adopted in 2012.

Conformance Verb	Cadinality	Result	Schematron Result
SHALL	m..n, where m > 1 and n ≥ m Forms 1..1 1..* 1..some number	the element or attribute must be present at least m and no more than n times	If the context node **a** is present, and the specified node **b** is absent, or is present more than n times, validation ought to yield an error message. The lower bound of 1 indicates to the human reader that absence is an absolute error. Examples[1]

SHOULD	0 .. n, where n ≥ 0 Forms 0..1 0..* 0..some number	the element or attribute should be present at least 1 times and no more than n times	If the context node is present, and the specified node is absent, or is present more than n times, validation ought to yield a warning message. The lower bound of 0 indicates to the human reader that absence is not an absolute error.
MAY	0 .. n, where n ≥ 0 Forms 0..1 0..* 0..some number	the element or attribute may be present at least 1 times and no more than n times	If the context node is present, and the specified node is absent, validation ought to yield no messages. If the context node is present, and the occurrences of the specified node are greater than n, ...? The lower bound of 0 indicates to the human reader that absence is not an absolute error.
SHALL NOT	0..0	the element or attribute is not allowed	If the context node is present, and the specified node is present, validation ought to yield an error message.
SHOULD NOT	0..0	the element or attribute should not be present, but may be	If the context node is present, and the specified node is present, validation ought to yield a warning message.

Examples[1]

Scenario

An observation template has the constraint SHALL 1..1 effectiveTime

Result

When the context node is present (the observation) and effectiveTime is not present in the context node, validation will produce an error message.

Scenario

A "SHOULD effectiveTime" has the constraint "SHALL 1..1 high"

Result

When the context node is present (the effectiveTime), and high is not present within the context node, validation ought to produce an error message.

Note: isBranch constraints cannot be evaluated line by line as above. At least for SHALL constraints they must be evaluated as a whole.

Scenario

isBranch SHALL in the form "SHOULD 0..1 effectiveTime such that it SHALL 1..1 high"

Result

The parent of this branch/slice ought to throw an error if it contains no effectiveTime/high.

Other effectiveTime elements that contain no high element are not precluded by this constraint.

Bindings

See CDA Best Practices and FHIR Best Practices for more information

Single-Value Binding

A single-value binding is when you require that the element/attribute in the template/profile always have the same (pre-defined) value.

Example #1: SHALL contain 1..1 statusCode="completed"

Example #2: SHALL contain 1..1 code="42348-3" Advance Directives (CodeSystem: LOINC urn:oid:2.16.840.1.113883.6.1)

Value Set Binding

A value set binding is when you associate an element/attribute with a (already-defined) value set. A value set specifies a set of codes drawn from one or more code systems.

Specifying a value set allows the implementer of the template/profile to choose a code within the value set that is appropriate for the scenario.

Example: SHALL contain 1..1 code, which SHOULD be selected from ValueSet AdvanceDirectiveTypeCode urn:oid:2.16.840.1.113883.1.11.20.2 STATIC 2006-10-17

Code System Binding

A code system binding is when you associate an element/attribute with a (already-defined) code system. A code system is external terminology or ontology such as LOINC , or SNOMED CT.

Specifying a code system allows the implementer of the template/profile to choose a code from the code system that is appropriate for the scenario; such as allowing the implementer to choose any code from LOINC.

Example: SHALL contain 1..1 code, which SHOULD be selected from CodeSystem LOINC (urn:oid:2.16.840.1.113883.6.1)

Categories

Associate One or More Categories to a Constraint

1. Open the **Template Editor** for the desired template
    - The implementation guide associated with the template/profile must have categories defined. See Authoring > Implementation Guide > Categories for more info.
2. Go to the **Constraints** tab
3. Select a constraint
4. In the Edit Constraint pop-up window, a multi-select list of categories will display, showing each of the categories associated with the implementation guide that the template/profile is associated with
5. Select on a category to assign it to the constraint
    - Hold CTRL while selecting to select multiple categories for the constraint
6. Save the template

Choice Elements

Some elements will show (for example) "effective[x]". These, elements are called "choices". These choice elements indicate that only one of its children may be used at a time. For example:

- effective[x]
    - effectiveDateTime
    - effectivePeriod

In this case, an instance of the template/profile may only contain *either* effectiveDateTime or effectivePeriod.

Valid XML Example:

```
<Observation>
  <effectiveDateTime value="xxx" />
</Observation>
```

Invalid XML Example:

```
<Observation>
  <effectiveDateTime value="xxx" />
  <effectivePeriod>
    <from value="yyy" />
    <to value="zzz" />
  </effectivePeriod>
</Observation
```

The Template/Profile editor *does* allow you to choose/constrain more than one option; effectively giving the implementer the choice to choose which option for themselves.

Template/Profile Editor

The template/profile editor allows you to specify conformance and cardinality for the choice as a whole (ex: "effective[x]") but does not allow you to specify conformance/cardinality for the options within the choice (ex: "effectiveDateTime" and "effectivePeriod").

Narrative Generation

Here is an example of narrative generated for choice elements when multiple options are constrained/selected:

1. **MAY** contain zero or one [0..1] **onset[x]**, where **onset[x]** is one of (CONF:3272-43)
    1. **onsetDateTime** (CONF:3272-44)
    2. or **onsetAge** (CONF:3272-45)

Preview

As changes are made to constraints within the template, the “Preview” tab allows the user to see the format of those changes in the final export. Heading levels and descriptions are included in the preview.

Validation

Upon saving a template, the Validation tab shows recommendations and warnings based on the constraints defined in the template. The following rules are taken into consideration:

- **Template context is not found within the schema**. In earlier versions of Trifolia (prior to version 2.0),

users could free-text the template context-type. If the context type was spelled incorrectly, the schema validation would fail. Template context validation could also fail if we changes/updates are made to the schema, and the portion of the schema that the template represents no longer exists. One example of this is with FHIR profiles. If any resource types is removed, then the context of templates built on that resource type would not be able to be found in the schema.

- **Custom Schematron Syntax**. Trifolia compiles all custom Schematron when the template/profile is saved and determines if the Schematron is syntactically correct.
- **Primitive constraint with no prose**. Occurs when creating a primitive constraint but not specifying the constraint prose. Primitive constraints without any prose are ignored during export.
- **Constraint's context not found in schema**. This can occur in templates that were designed using legacy versions of Trifolia, which allows template/profile authors to freely edit the context of the constraint. This can also occur when a template/profile has been moved from one type/context to another type/context (ex: section to entry) and the constraints within the template/profile have not been reconciled to the new type's structure.
- **Branch/slice without identifiers**. At least one identifier should be present within every branch/slice to indicate how the one branch/slice is different from other branches.
- **Invalid contained template**. The context of a contained template/profile does not match the data-type of the constraint containing the template.
- **Invalid constraint cardinality**. The constraint loosens the cardinality requirements from the base standard/schema (ex: base schema requires 1..1 code and a constraint in the template/profile indicates 0..1 code).

Publish Settings

Use the Publish Settings page to control the way a template/profile appears in an implementation guide.

You can control the following:

- Generate and customize code samples to associate with the template.
- Control whether one or more constraints has its own heading in the implementation guide.

View the Publish Settings page

Select Publish Settings from the Template Viewer's Quick Buttons.

Constraint Headings

For complex template/profile structures, it is best practice to document certain elements of the template/profile in greater detail.

Use the Constraints tab of the Publish Settings page to add headings to constraints that appear in an implementation guide. This allows complex templates to be broken into sections within the implementation guide.

Add a Heading to a Constraint

1. In the Constraints section of the Publish Settings page, select Edit for the constraint you want to edit.
2. To add to the constraint heading text, select the Heading check box. The Heading Description box appears.
3. Enter text in the Heading Description, Description, and Label boxes. The text appears as shown in the example below.
4. To add an inline code sample, select Add. The Edit Constraint Sample window appears. Enter a Name for the sample, and enter the sample code in the Sample Contents box, then select OK. The sample name appears in the sample list.
5. When finished, select OK. Heading text, Description text, Label text, and Sample Code are added to

the selected constraint. The effect is shown below.

6. When finished, select Save in the Edit Publish Settings.

Samples

Use the **Template Samples** tab of the **Publish Settings** page to generate, customize, and format code samples to associate with a particular template.

Generate a Code Sample

1. In the Template Samples tab, select Add. The Edit Sample page appears.
2. Enter a Name for the sample.
3. Enter or paste the text you want to use in the sample, or select Generate to create a code sample based on the template's constraints.
4. If you generated a sample, edit it as needed.
5. To format the sample automatically, applying standard indentation, Format and Indent.
6. When finished, select OK. The sample is saved, and is available for export with the implementation guide.

Copying

TODO

Moving

A template/profile needs to be "moved" when you need to change fundamental traits of the template/profile, such as the "type" or the implementation guide that it is associated with. Trifolia uses the implementation guide's type (CDA vs. FHIR, etc.) as well as the template/profile's type (document, section, Composition, Person, etc.) to determine what portion of the base standard the template/profile represents. When this fundamental information is changed, the nodes in that portion of the base standard may be different, and constraints may need to modified for the template/profile to be valid. The "move" screen gives you the opportunity to identify these differences, and decide what changes should be made on the template/profile.

Steps to move a template/profile

1. Select "Browse" > "Templates/Profiles".
2. Find the template/profile in question, and select "Edit". The "Template/Profile Editor" will appear.
3. On the first "Meta-Data" tab, click the "Move" link on the right side of the screen.
4. Select the new implementation guide, type and what element the template/profile applies to.
5. Click "Next".
6. The screen appeared next will show you the elements/attributes that are available with the new selection, valid constraints in bold, and highlights elements/attributes in *red and italics* that are no longer available.
7. You *should* choose to remove all of the constraints that are no longer available. If you choose to keep any of these invalid constraints, the template/profile will not be valid; which may result in broken exports.
8. After you have removed invalid constraints, click the "Finish" button. You will be returned to the "View" screen, and the template/profile will be moved.

Deleting

Deleting a template/profile is permanent, and cannot be undone. Templates/profiles that are referenced *by* the template/profile being deleted will not be affected. Templates/profiles that reference the template/profile

being deleted will have their constraint changed so that the reference no longer exists.

Deleting a template/profile may have unexpected downstream impacts in the implementation guide. For example, if you delete a template/profile that makes reference to a template/profile from another implementation guide, the template/profile from the other implementation guide will not appear in the implementation guide's MS Word and HTML exports.

Steps to delete a template/profile

1. Select "Browse" > "Templates/Profiles".
2. Find and click "View" on the template/profile being deleted.
3. Click the "Delete" button at the top navigation bar.
4. The screen shown next indicates how many templates/profiles reference the template/profile being deleted.
    1. Specify a template/profile to replace the reference. This will update the other templates/profiles that reference the template/profile being deleted to reference the specified template/profile.
5. Additionally, this screen shows how many samples are directly associated with the template/profile being deleted.
    1. These samples will be permanently deleted.
6. Click "Delete".
7. You will be returned to the "Browse" > "Templates" screen once the template/profile is deleted.

Terminology

Edit a Value Set

1. In the Value Sets tab, locate the value set you want to edit, or select Add Value Set to add a new set. The Edit Value Set screen appears.
2. Enter the value set's meta data as described below.
3. For each member of the value set, select Add Member.
4. When finished, select Save. The value set is saved.

Value Set Meta Data

- **Name** - Enter a name for the value set.
- **Identifier** - Enter an identifier for the value set in one of two formats
    - urn:oid:XXXX
    - http[s]://XXXX
- **Code** - Enter a code value set.
- **Intensional** - Indicate whether the value is computable.
- **Intensional Definition** - Describe how to compute the value set. You may want to use custom scripting; the ultimate goal is to demonstrate that “this value set references Value Sets X and Y”
- **Description** - A description of the value set and its purpose.
- **Source URL** - Indicate the source of this value set.
  For example: http://www.phconnect.org/group/phinvads (CDC's Public Health Information Network Vocabulary Access and Distribution System)
- **Incomplete** - Check this box if the value set is not yet complete.

Value Set Member Data

- **Code** - Enter a code for the value.
- **Display Name** - Enter a display name for the value.
- **Code System** - Choose the code system that contains value set.
- **Status** - Indicate whether the value is currently active or inactive.
- **Status Date** - Enter the date the status was changed.

Edit a Value Set's Codes / Concepts

1. In the Value Sets tab, locate the value set you want to edit.
2. Select the menu drop-down on the value set.
3. Select Edit Concepts. The Edit Value Set Concepts screen appears.
4. Add concepts by completing the fields in the bottom row of the top table and selecting Add. After adding a concept, the concept will move to a new table called"Pending Changes.
5. Edit concepts already in the value set by finding the concept in the top table and selecting the Edit button. The values in the columns of the first table will change to represent editable fields. When done making changes, select the OK button to the right of the row. The changes made to the concept are reflected in the Pending Changes table.
6. Remove an existing concept from the value set by finding the concept in the first table and selecting the Remove button to the right of the row. The concept will be moved to a Pending Removal table at the bottom of the screen.
7. When done adding, editing and removing concepts in the value set, select the Save button at the bottom of the screen to persist the changes.

Edit a Code System

1. In the Code System tab, locate the code system you want to edit, or select New to add a new code system. The Add/Edit Code System window appears.
2. Enter the code system information as described below.
3. When finished, select Save. Your changes are saved.

Code System Meta Data

- **Name** - Enter a name for the code system.
- **Identifier** - Enter the code system's identifier in one of the following formats:
    - urn:oid:XXXX
    - http[s]://XXXX
- **Description** - Enter a description for the code system.

Delete a Value Set

1. From the Value Sets tab of the Terminology Browser, locate the value set you want to delete.
2. Select the drop-down menu arrow to the right of the value set name and choose Remove. A confirm message appears.
3. Select OK. The value set and its members are removed.

Delete a Code System

1. From the Code Systems tab of the Terminology Browser, locate the code system you want to delete.

2. Select the drop-down menu arrow to the right of the code system name and choose Remove. A confirm message appears.
3. Select OK. The code system is removed.

MS Word

Export Templates/Profiles

1. Click the "Export" button from the navigation menu at the top of the screen.
2. Select the implementation guide you wish to export
3. Select the "MS Word" format from the drop-down
4. In the **Content tab**, choose the content you want to include in the export file. You can also choose a sort order for the templates.
5. In the **Value Sets tab**, choose whether to include value sets in the export file, the maximum number of values to include, and where they are located in the export file.
6. In the **Templates tab**, choose the templates you want to include. By default, all templates are selected. You can also choose to Include Inferred templates.
7. Select Export after the implementation guide has finished validating and loading.
8. You will be prompted to download the DOCX file produced.

Export Settings

Each tab of the Export Templates to MS Word page contains settings specific to the tab.

The Content Tab

- **Sort Order** - choose Alphabetically to sort by template/profile name, or Alpha-Hierarchical to sort by template/profile name within the referenced template/profile hierarchy. Templates are sorted first by type, then by template/profile relationship, then alphabetically within that relationship. This is much easier to understand if you look at the Containment table in the export document, as described in the following section.
- **Document Tables** - choose from the drop-down menu to include the Template List table, the Template Containment table, or Both.
- **Template Tables** - choose from the drop-down menu to the Context table, the Constraint Overview table, or Both.
- **XML Samples** - select the Include check box to include XML samples in the export file.
- **Change List** - select the Include check box to include the Change List in the export file.
- **Publish Settings** - select the Include check box to include Publish Settings in the export file.
- **Notes** - select the Include check box to include Notes in the export file.

Note: For more information about how vocabulary information is represented in by Trifolia template, vocabulary, and Schematron exports, see Terminology & Trifolia.

The Value Sets Tab

- **Tables?** - choose whether to include value set tables in the export file.
- **Maximum Members** - specify the maximum number of members that should be exported for a given value set.
- **Create as Appendix** - choose whether the value sets are included as an appendix.

Each value set included in the implementation guide may have a separate "maximum members" setting.

When changing the "Maximum Members" setting at the top of the "Value Sets" tab, all individual value sets are defaulted to the same "Maximum Members" value. After selecting a default for all value sets, you can change individual value sets to reflect a different "maximum members" value that is more appropriate for the individual value set.

The Templates Tab

The Templates tab shows a list of templates that appear in the selected implementation guide.

- **Parent Template** - choose a template/profile to export only a subset of the templates in a guide. The export file includes the parent template/profile and any templates referenced by it. The Include Inferred check box includes all referenced templates; if not selected, the export file includes only those templates contained in the implementation guide.
- **Include Inferred** - choose whether to include inferred templates in the export file. For larger guides, this may increase generation time and may result in a significantly larger document.
- **Template List** - select or clear check boxes to select the templates you want to include in the export file. Select or clear the check box in the table's header row to select or clear the boxes for all templates.
- **This IG? column** - Yes indicates that the template/profile is contained in the implementation guide. No indicates that the template/profile is referenced by one of the templates in the implementation guide, but is not contained in it.

Document Tables

You can choose to include two automatically-generated tables in the export document:

- **List table** - lists the templates in the implementation guide.
- **Containment table** - list the templates within their referenced hierarchy.

To choose the document tables to include:

- Choose **None** to exclude both tables.
- Choose **Both** to include both tables.
- Choose **List** to include only the List table
- Choose **Containment** to include only the Containment table.

Template Tables

For each template, the default setting is “both” to export context and constraint-overview tables in the document export. However, you can choose to include none, one or both automatically-generated tables.

- **Context table** - lists the templates that use this template, and the templates that are used by it.
- **Constraint Overview table** - provides an overview of all constraints in the template, with a link to each constraint.

To choose the template/profile tables to include:

- Choose **None** to exclude both tables from each template.
- Choose **Both** to include both tables in each template.
- Choose **Context** to include only the Context table in each template.
- Choose **Containment** to include only the Containment table in each template.

Save Default Export Settings

Users that have "Edit" permission for the implementation guide being exported have the option to save the current configuration settings as the default settings for all users. For these permitted users, a checkbox is available at the bottom of the "Export Templates to MS Word" screen which, when checked, saves the current configuration of settings as the default settings. When any user exports the same implementation

guide's templates to MS Word, these default settings will be used. Users that do not have edit permissions to implementation guide will still be able to make their own customizations to the settings; however, they will not be able to store them as the default settings.

Web-Based HTML IG

Trifolia can produce an HTML package that can be viewed in any browser, or hosted on a web server to be accessible on-line. An example of a HTML implementation guide can be found here: http://lantanagroup.github.io/trifolia/Public_Testing_IG/index.html

The HTML implementation guide includes all of the same information as the MS Word format. The HTML implementation guide includes some additional features and functionality:

- Overview information
- Searching
- UML relationship diagram for each template/profile
- Complete code listing for value sets
- Tabbed *and* panel views
- Colored and formatted samples

Steps to view an HTML implementation guide

1. Go to "Browse" > "Implementation Guides"
2. Select "View Web" button next to the implementation guide you would like to view

Download the HTML implementation guide while viewing

1. View the HTML implementation guide (as described in "Steps to view an HTML implementation guide").
2. In the top-right of the screen, click the "Download" button.
3. When prompted, download the .zip package to your computer. This zip package contains all of the files necessary to view the HTML implementation guide without a connection to the internet.

Export the HTML implementation guide

1. Click "Export" from the top navigation menu
2. Select the implementation guide to export
3. Select "HTML" as the export format in the drop-down menu
4. Click the "Export" button after the implementation guide has finished validating and loading
5. You will be prompted to download the .zip package produced

Implementation guide snapshots

With the XML/JSON export, you can download a JSON snapshot of all data needed for the HTML implementation guide. Trifolia allows you to upload that snapshot to the implementation guide so that you can view that point-in-time representation of the HTML implementation guide in the future. After uploading the exported snapshot back to Trifolia as a "Data Snapshot", the "View Implementation Guide" screen shows a new tab called "Web Publications". Each snapshot uploaded represents a separate "Web Publication". Clicking one of the web publication links will open the HTML implementation guide representing that snapshot in a separate browser window/tab.

Steps to create a web publication from a snapshot

1. Using the "Export" > "XML/JSON" screen, export the implementation guide in the "Data Snapshot (JSON)" format

2. Using the Browse > "Implementation Guides" screen, view the implementation guide
3. Click the "Edit" drop-down, and select "Files"
4. Click "Add file" to upload the snapshot previously exported
5. For "File", select the snapshot downloaded previously
6. For "Type", select "Data Snapshot (JSON)"
7. Provide a description
8. Click the "OK" button
9. Click the "Save" button
10. Return to the "View" implementation guide screen, and you will see a new "Web Publications" tab that provides a link to view the HTML implementation guide representing the point in time which the snapshot was exported

Schematron

Export Schematron

1. Click "Export" from the top navigation menu
2. Select the implementation guide to export
3. Select "Schematron" as the export format
4. In the **Options** tab, choose the **Value Set Format**, specify the **Value Set File Name** to use for value set output, and choose whether to use custom Schematron.
5. In the **Templates/Profiles** tab, select or clear check boxes to indicate the templates you want to include in the export file. Select or clear the check box in the table's header row to select or clear the boxes for all templates.
6. Select **Export** after the implementation guide has validated and finished loading
7. You will be prompted to download either a .zip or .sch file, depending on whether "Include Vocabulary" was selected

The Options Tab

- **Value Set Format** - choose the format you want to use from the drop-down menu
- **Include Custom Schematron** - choose whether to include any custom Schematron entered on the Edit implementation guide > Custom schematron screen.
- **Value Set File Name** - enter a new filename to use for value set output, or use the default, voc.xml.
- **Default Schematron** - specify the assertion that should be used when a primitive constraint is defined that does not have custom schematron defined for it.
- **Include Vocabulary** - when "yes" is selected, the export produced is a ZIP file, instead of a single SCH file. The ZIP file includes the Schematron (SCH) file, as well as a vocabulary XML file used by the Schematron.

The Templates/Profiles Tab

When "Include Inferred" is "Yes", all templates/profiles owned by the implementation guide and all templates referenced by templates within the selected implementation guide will be included in the list of templates. When "Include Inferred" is "No", only templates owned by the selected implementation guide will be included in the list of templates. See Common Features > Inferred Templates for more information.

Select or clear check boxes to choose the templates you want to include in the guide. A **No** in the **This IG?** column indicates that the template/profile is referenced by, but not contained in the implementation guide. Select or clear the check box at the top of the column to select or clear all check boxes.

Terminology

Export Vocabulary

1. Click "Export" from the top navigation menu
2. Select the implementation guide to export
3. Select one of the "Vocabulary" formats in the drop-down menu
4. Choose a Format for the export. Vocabularies can be exported in these formats:
    1. Lantana standard (SCH)
    2. Sharing Value Sets (SVS)
    3. Excel (XSLX)
5. In the Maximum Members box, enter or use the arrow buttons to set the number of members a value set must contain in order to be excluded from the export.
6. Choose an Encoding format.
7. Select Export after the implementation guide has validated and finished loading
8. You will be prompted to download the .xml or .xlsx vocabulary file, depending on the vocabulary format selected.

The "Value Sets" tab on the export settings screen shows the user the value sets that will be included as part of the export. No changes can be made to this tab, it is only for informational purposes.

Export Formats

Value sets that are **dynamically** bound to an implementation guide (via a template's constraint) are only included in the following export formats:

- Excel
- FHIR

Lantana Standard (XML)

The "Lantana Standard" XML format is a proprietary format developed by Lantana. This format is used by schematron for validating value set bindings.

Sharing Value Sets (SVS/XML)

The Sharing Value Sets (SVS) profile provides a means through which healthcare systems producing or consuming clinical or administrative data, such as diagnostic imaging equipment, laboratory reporting systems, primary care physician office EMR systems, or national healthcare record systems, can access value sets built from common, uniform nomenclatures managed centrally. Shared nomenclatures with specific derived value sets are essential to achieving semantic interoperability.

Download Specification

Excel (XLSX)

The Excel format includes two sheets:

- Affected Value Sets
    - **Value Set Name**
    - **Value Set OID**
- Value Set Members
    - **Value Set OID**
      This is the identifier of the value set that the memeer belongs to (a reference to a row on the

"Affected Value Sets" sheet)

- **Value Set Name**
  This is the name of the value set that the member belongs to (a reference to a row on the "Affected Value Sets" sheet)
- **Code**
- **Display Name**
- **Code System Name**

FHIR

The output of this export is in FHIR DSTU1 XML format. Additional details on this format can be found here.

XML/JSON

Export Templates to XML/JSON

1. Click "Export" in the top navigation menu
2. Select the implementation guide to export
3. Select one of the XML or JSON formats in the drop-down menu
4. Choose the templates you want to include in the guide. You can select a parent template/profile to include only the templates associated with that template.
5. Select Export after the implementation guide has validated and finished loading
6. You will be prompted to download the .xml or .json export of your implementation guide

XML Type

- **Trifolia XML**: This is an XML format for templates proprietary to Trifolia. The schema for this format can be downloaded here.
- **FHIR XML**: This option is only available for FHIR implementation guides. This converts the templates in the implementation guide into the XML format defined by FHIR DSTU 1. This XML export includes both FHIR Profile and ValueSet resources. See http://www.hl7.org/fhir for more details on FHIR DSTU 1.
- **FHIR Build**: This option is only available for FHIR implementation guides. This create a zip package that can be used by the FHIR IG publisher to produce HTML files following the same styles and practices as the core FHIR standard.
- **Data Snapshot**: This is a JSON-based export of the implementation guide (including all templates/profiles, value sets, code systems, volume 1 content, etc). It can be uploaded to the "Files" section of an implementation guide and used as a point-in-time snapshot with the web-based IG.

FHIR

Importing using FHIR REST interface

Trifolia provides *some* (but not all) POST/PUT/DELETE functionality for FHIR profiles. Below is a list of the operations supported by Trifolia:

- FHIR DSTU2 /api/FHIR2/
  - POST /api/FHIR2/StructureDefinition
  - PUT /api/FHIR2/StructureDefinition/{id}
  - DELETE /api/FHIR2/StructureDefinition/{id}

    - POST /api/FHIR2/ValueSet
    - PUT /api/FHIR2/ValueSet/{id}
- FHIR STU3 /api/FHIR3/
    - POST /api/FHIR3/StructureDefinition
    - PUT /api/FHIR3/StructureDefinition/{id}
    - DELETE /api/FHIR3/StructureDefinition/{id}
    - POST /api/FHIR3/ValueSet
    - PUT /api/FHIR3/ValueSet/{id}

Native

Importing using the native XML format with the web interface

Trifolia can import implementation guide and template/profile data via the "Import" tab. The Trifolia native XML format includes all information that Trifolia uses to generate various exports, including sample information for both templates/profiles and constraints, heading information, template type descriptions, etc. This export/import format is designed to transfer information between different installations of Trifolia.

To import this format:

1. Select the "Import" tab from the top navigation menu
2. Select the file to import
3. Click the "Import Now" button

After the import is complete, you will be presented with a report indicating what was changed (or not changed) during the import. Templates/profiles shown the report can be expanded to see what constraints and samples were modified.

Here are some key things to know when importing data using this method:

- The import *does* overwrite existing information..
- If the import includes implementation guides or templates that already exist, and changes are made to the implementation guides and templates/profiles, you must have access to edit the implementation guide and/or template, or the import will fail.
- If the import includes new implementation guides or templates that do not exist, then as long as the user has a role that allows for the creation and editing of implementation guides and/or templates, the import will be successful.
- Templates/profiles, constraints and samples that already exist and are not included in the import XML file are *not* deleted. For example, if Trifolia already has templates/profiles A, B and C in "My Implementation Guide", and the import XML file only contains templates/profiles A and B, then template/profile C will still remain in "My Implementation Guide" after the import is complete.

Terminology

Importing Value Sets

The Browse > Terminology screen allows you to import value sets from either VSAC or PHIN VADS. Imported value sets cannot be edited/modified using Trifolia.

1. Select Browse > Terminology
2. Click the "Import Value Set" button in the top-right
3. Specify the source from where you want the value set imported
4. Enter the identifier (in most cases, the OID) of the value set to import

5. Click OK

If the value set changes and needs to be re-imported into Trifolia:

1. Click the link on the value set in Browse > Terminology
2. Select "Re-Import"
3. Click "OK" on the pop-up window for importing

Importing from VSAC

When Trifolia imports a value set from the VSAC, it always imports the latest version of the value set. This may be enhanced in the future to allow specifying which version of the value set to retrieve.

The identifier created for the value set by trifolia is an HL7-II identifier type, which includes the OID of the value set and the version/date of the value set. For example: urn:hl7ii:2.16.840.1.113883.3.2288:20170320 where 2.16.840.1.113883.3.2288 is the OID of the value set, and 20170320 is the version/date of the value set.

Credentials/Licensing

See Getting Started > User Profiles for more information.

Reports

Use the Reports menu to choose and generate a variety of reports. The following reports are available from the Reports menu.

- **Template Review** - lists templates and their details. Filter and group in many ways.
- **Template Validation** - select the implementation guide you want to validate. Select to show Warnings, Errors, or All. Shows the status of any templates containing warnings or errors.
- **Organization** - lists all users in an organization, along with their permissions and contact info.

View a Report

1. From the Reports menu, choose the report you want to view. A report page appears, with a page summary and navigation at top. A filter box appears at the top of each column.
2. Enter filters in the top row of the report to limit the data that appears in the report as described in Report Filters, below.
3. Print the report as needed.

Report Filters

You can filter by the contents of a column to limit the data that appears in the report.

- **String filter** - Enter text in one or more columns.
- **Dropdown filter** - Choose a filter from the drop-down menu at the top of a column to view only entries with that value.

FHIR

Trifolia's implementation of the FHIR specification is documented here. This topic includes information on the latest implementation of FHIR in Trifolia; previous versions are kept for backwards compatibility, but are not actively maintained. **Trifolia currently supports the STU3 version of FHIR.**

The following table(s) indicate what properties Trifolia uses of the FHIR specification, and any relevant notes about the property's usage in Trifolia.

StructureDefinition

Property	Notes
id	
name	
description	
url	
type	
context	
contextType	
derivation	
extension	
contact	Not imported with POST/PUT. Exported with GET, represents the author of a profile.
baseDefinition	
differential	
snapshot	Not imported. Only differential definitions are imported.

ElementDefinition

Property	Notes
id	
short	Constraint's label if specified, or the element/attribute name if not
label	Constraint's label if specified
comment	Constraint's notes, if any
path	
min	
max	
slicing	When the constraint, or a parent constraint is a branch/slice
slicing/discriminator	The first "identifier" within the branch/slice or "@type" when constraint is a schema-based choice
slicing/rules	Always set to "open" when constraint is a schema-based choice
binding[x]	
binding/strength	
bindingReference/reference	
bindingReference/display	
pattern[x]	

patternCodeableConcept	When there is a single-value binding and the data-type is CodeableConcept
patternCoding	When there is a single-value binding and the data-type is Coding
patternCode	When there is a single-value binding and the data-type is code
patternString	When there is a single-value binding and the data-type is not one of the above.
type	
type/code	"Extension" when the profile is for an Extension, otherwise "Reference"
type/profile	Contained profile's "url", when profile is for an extension
type/targetProfile	Contained profile's "url", when profile is not an extension

ImplementationGuide

TODO

ValueSet

TODO

API

Trifolia includes limited functionality for the FHIR REST API.

Versions

Trifolia supports three different versions of the FHIR REST API for backwards compatibility. However, only the latest version is actively developed and improved.

Each of the base endpoints correlate to their version of the FHIR standard:

Version	**Endpoint**	**Notes**
DSTU2	/api/FHIR2	No longer maintained
STU3	/api/FHIR3	Issues maintained upon request
Current Build	/api/FHIRCurrentBuild	Current, actively maintained

Additionally, each of these endpoints correlate to the appropriate Implementation Guide Type in Trifolia. For example, only Implementation Guides associated with the Implementation Guide Type "FHIR Latest" will be returned by requests to the /api/FHIR3 endpoint. Similarly, only Implementation Guides associated with the Implementation Guide Type "FHIR DSTU2" will be returned by the /api/FHIR2 endpoint.

Authentication

All REST endpoints are authenticated with OAuth 2.0 using a Authentication header with a "Bearer" token.

To get a token to use for authentication, login to Trifolia and click on the "My Profile" menu under your name in the top-right corner. In the "My Profile" page, the authentication token for your logged-in user will be shown in a separate panel.

Use the authentication token in the "My Profile" page in REST API calls by adding an "Authorization" header with a value of "Bearer <TOKEN>".

Customizations

- StructureDefinition.id cannot exist in the Profile when performing a POST

- StructureDefinition.id must not be changed when performing a PUT

Supported Operations

See Developer Docs for more details.

Method	Endpoint
GET	/api/FHIR3/metadata /api/FHIR3/CapabilityStatement
GET	/api/FHIR3/ImplementationGuide /api/FHIR3/ImplementationGuide/_search
GET PUT DELETE	/api/FHIR3/ImplementationGuide/{implementationGuideId}
POST	/api/FHIR3/ImplementationGuide
GET	/api/FHIR3/StructureDefinition /api/FHIR3/StructureDefinition/_search
GET PUT DELETE	/api/FHIR3/StructureDefinition/{id}
POST	/api/FHIR3/StructureDefinition
GET	/api/FHIR3/ValueSet /api/FHIR3/ValueSet/_search
GET PUT	/api/FHIR3/ValueSet/{id}
POST	/api/FHIR3/ValueSet

Extensions

Re-usable Extensions

Trifolia supports re-usable extensions for FHIR DSTU2 implementation guides and profiles.

Creating a Re-usable Extension

1. Create a new profile in a FHIR DSTU2 implementation guide.
2. Select FHIR DSTU2: Extension for the "type" of profile.
    1. Note that the Applies To button is disabled for Extension profiles.
    2. After selecting Extension as the type of profile, a constraint is automatically created by Trifolia for the "@url" attribute and matches the identifier specified for the extension. Any time the identifier is updated, the constraint will automatically be updated as well.
3. Define a constraint for the type of value your extension will require (such as "valueCodeableConcept").
4. Save the profile.

Trifolia treats all profiles that are of type FHIR DSTU2: Extension as re-usable extensions. The behavior of

Trifolia's template/profile editor adjusts slightly (such as disabling the Applies To selection button, and automatically adding a @url constraint) to account for requirements in the core FHIR DSTU2 standard when editing extension profiles.

Using a Pre-defined Extension

1. Open any FHIR DSTU2 profile.
2. In the constraint editor, select an "extension" element that does NOT already have a constraint defined for it.
3. A drop-down list shows in the constraint editor panel that provides a list of all extensions for which you have permissions.
4. Select the extension you want to add to the profile and select the "+" (add) button.
5. The "extension" element is turned into a constraint, and automatically creates constraints for each of the constraints defined in the extension.
6. The "extension" element itself should be marked as a slice and the @url constraint within the extension should be marked as a descriminator. The conformance and cardinality is set to "SHALL 1..1" for the new extension constraint, as well as the @url descriminator.

Profile Extensions

The profile may define extensions in addition to the constraints. These are extensions that add information to the profile as a whole, but are not required to be implemented where the profile is used/asserted.

Both FHIR DSTU1 and FHIR DSTU2 profiles support extensions on the profile.

Adding Extensions to a Profile

1. Edit any FHIR profile.
2. The bottom entry in the Extensions panel of the Template/Profile tab always represents a "new" extension.
3. Fill in the identifier, type, and value of the extension (all three fields are required).
4. Select the Add button to the right of the fields.
5. The extension is now added to the profile, and the profile can be saved. The extension added may be edited, but the type of the extension may not be changed; only the identifier and the value may be changed after extension is added.

Note: A future version of Trifolia will incorporate re-usable extensions in the profile extensions.

Developer Docs

This topic provides documentation to developers that would like to integrate with Trifolia or contribute to Trifolia's open source code-base.

API Help Documentation

Trifolia's API help documentation is automatically generated based on code-comments. The help documentation can be found at http://your.installation/api/Help.

Lantana's installation of Trifolia exposes the help documentation here:
https://trifolia.lantanagroup.com/api/Help

WADL (rest endpoint specification)

The WADL can be found here:http://your.installation/api/Help/Wadl

Lantana's installation of Trifolia exposes the WADL here: https://trifolia.lantanagroup.com/api/Help/Wadl

FAQ

In a CDA template, should I put single-value code bindings at the attribute level, or the element level?

The preferred way for all terminology bindings is to put them at the root level (i.e. the code element). The only time that you want to specify individual constraints at the attribute level is when you want to prohibit the use of nullFlavor.