Galaxy Tool Generator Documentation

Ming Chen, Bradford Condon

Nov 29, 2018

Contents:

CHAPTER 1

Installing GTG

1.1 Requirements

GTG depends on two Docker images: `statonlab/galaxy_tool_generator` and `bgruening/galaxy-stable:17.09`. First, you need to install Docker in your system. Then, run the following command to get the two images.

```
docker pull statonlab/galaxy_tool_generator
docker pull bgruening/galaxy-stable:17.09
```

1.2 Launch GTG with Docker

Run the code below to launch GTG. This will start a GTG application at http://127.0.0.1:8089/ and a Galaxy instance at http://127.0.0.1:8090/.

```
git clone https://github.com/statonlab/galaxy_tool_generator.git
cd galaxy_tool_generator && docker-compose up -d
```

To shut down GTG and the Galaxy containers:

```
docker-compose down
```

If you want to run GTG and the Galaxy containers at different ports, you can edit the port numbers in the *docker-compose.yml* file.

```yaml
version: '3'

services:
  app:
    image: statonlab/galaxy_tool_generator:latest
    ports:
      - "8089:80"
    volumes:
      - ./gtg_dev_dir/galaxy_tool_repository:/var/www/html/sites/default/files/galaxy_tool_repository
      - ./gtg_dev_dir/shed_tools:/var/www/html/sites/default/files/shed_tools
      - ./galaxy_tool_generator:/var/www/html/sites/all/modules/galaxy_tool_generator/galaxy_tool_generator
      - ./galaxy_tool_generator_ui:/var/www/html/sites/all/modules/galaxy_tool_generator/galaxy_tool_generator_ui
    command:
      - sleep 30
      - drush en -y galaxy_tool_generator galaxy_tool_generator_ui
  galaxy:
    image: 'bgruening/galaxy-stable:17.09'
    environment:
      - ENABLE_TTS_INSTALL=True
      - GALAXY_CONFIG_BRAND=GTG
    volumes:
      - ./gtg_dev_dir/shed_tools:/export/shed_tools
      - ./gtg_dev_dir/database:/export/galaxy-central/database
    ports:
      - "8090:80"
```

Port for GTG container

Port for Galaxy container

CHAPTER 2

Quick Start Guide

Note: Please see our detailed *User's Guide* for detailed instructions on using GTG.

- Open the GTG web interface.
- Use the **Create Tool XML** tab to start your XML file.
- Add XML components and set their attributes.
- Press the **Update XMLs in galaxy_tool_directory folder** button in the **Build Tool Repository** tab to add the finished XML to the repository.
- Add any additional files to the `gtg_dev_dir/galaxy_tool_repository` folder.
- Connect GTG to the Galaxy Toolshed in the **Connect to ToolShed** tab.
- Publish to the Test Toolshed in the **Publish Tool Repository** tab.
- Install and test your published tool in the local Galaxy container using the **Sync to Galaxy** field in the **Build Tool Repository** tab, providing the path relative to the `shed_tools` directory.
- Restart Galaxy to integrate the changes: `docker exec -it gtg_galaxy sh -c 'supervisorctl restart galaxy:'`

CHAPTER 3

User's Guide

3.1 Understanding the GTG workspace

After launching the GTG application, you should see the the folder `gtg_dev_dir` in your current directory and three subdirectories within it:

```
gtg_dev_dir/
├── database
├── galaxy_tool_repository
└── shed_tools
```

The `galaxy_tool_repository` subdirectory stores all files that form a Galaxy Tool Repository and can be published to Galaxy ToolShed with GTG. The subdirectory is mounted to the GTG container so that a developer can easily add non-XML files from the host machine to the GTG container. The XML files should be generated via GTG.

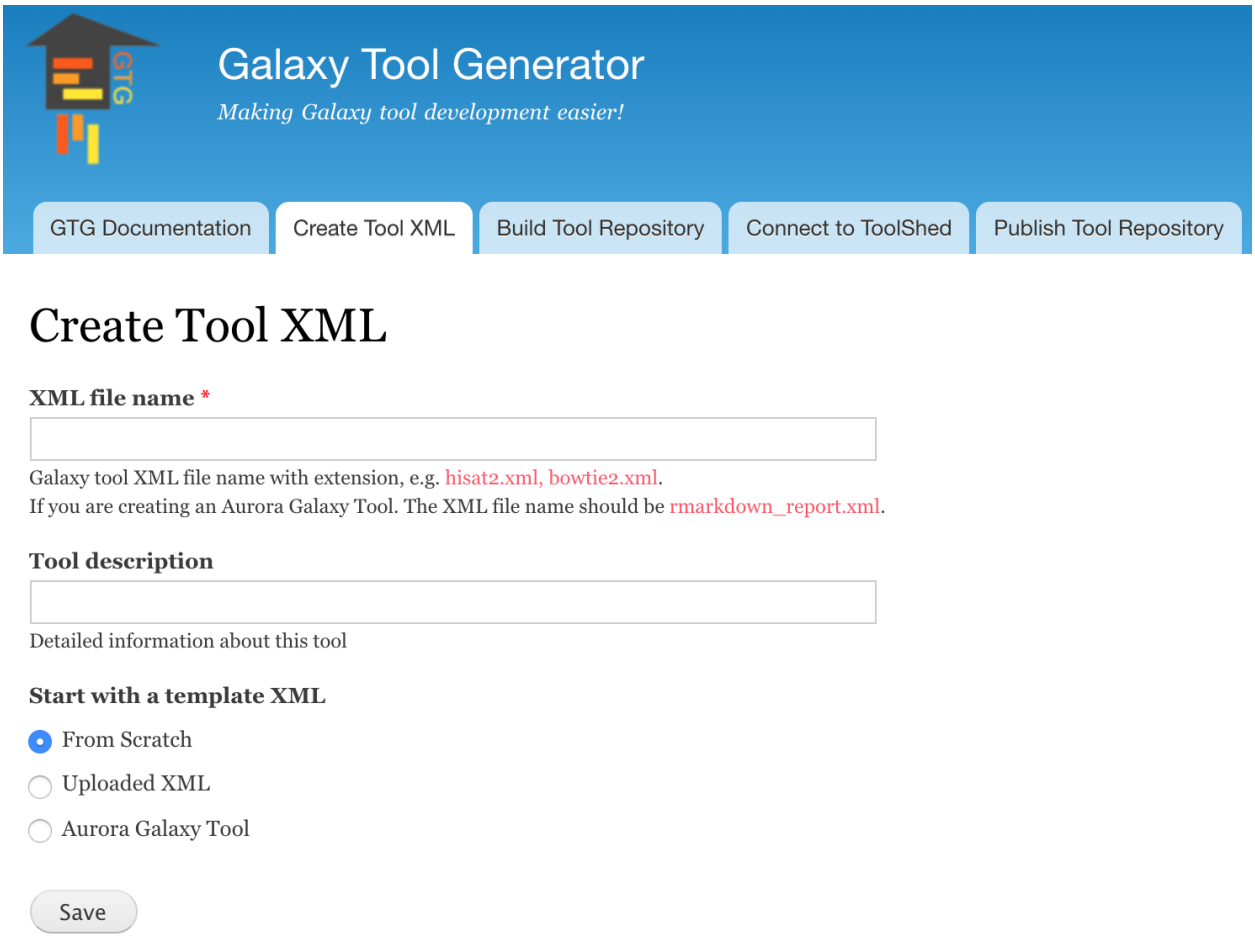
The `shed_tools` subdirectory is mounted to both the GTG container and the Galaxy container so that the galaxy tool repository being developed in GTG can be synced to the Galaxy instance for interactive testing.

The `database` subdirectory is mounted to the Galaxy container and displays the job working status of Galaxy. When the tool is being tested in Galaxy, the job running process can be monitored. This is useful for debugging your tools.

3.2 Creating the Tool XML

GTG provides three ways to build a Galaxy XML file:

- **From scratch**: builds XML from scratch using GTG.
- **Uploaded XML**: starts with an uploaded XML.
- **Aurora Galaxy Tool**: this option starts with an template file for developing an Aurora Galaxy Tool.



Select the appropriate method and click the **Save** button.

3.2.1 From Scratch

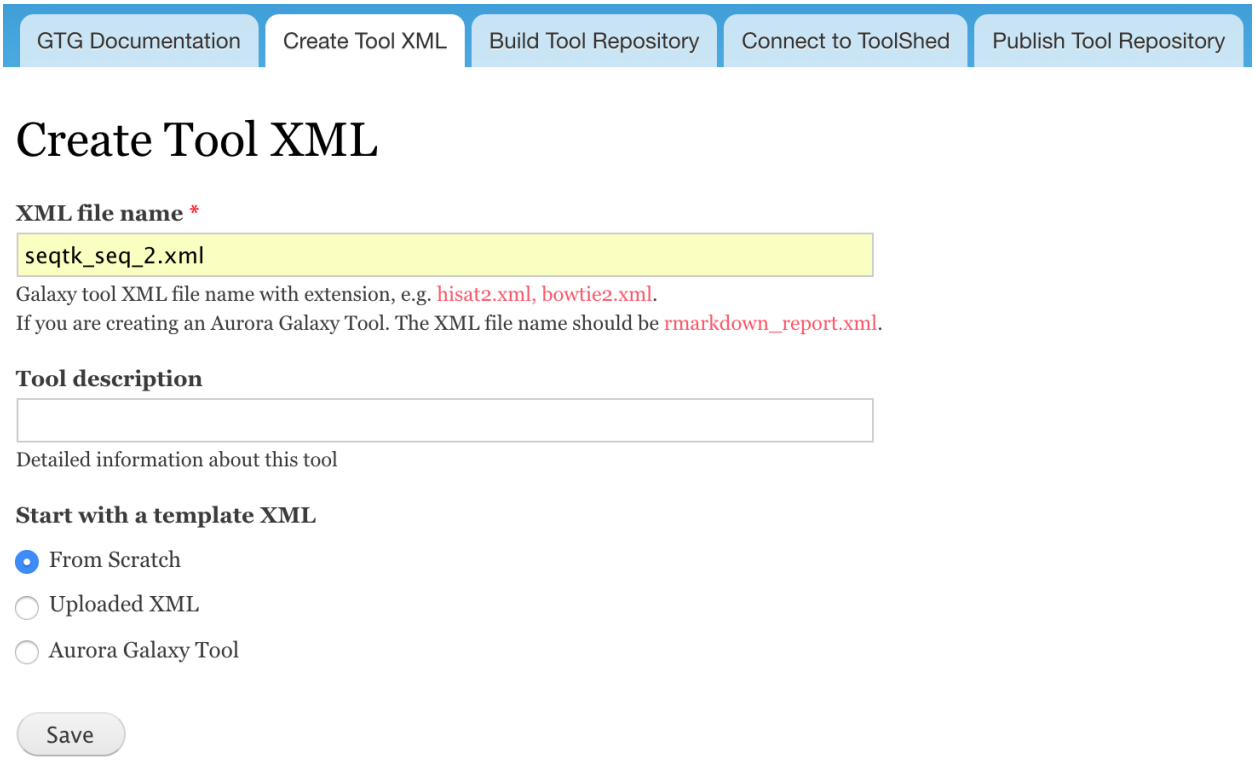
For comparison with another software for Galaxy tool development planemo, I am going to use an example from the planemo use cases. In this example we are going to use GTG to build this `seqtk_seq_2.xml` file.

In this guide, we'll create each piece of the XML, step by step, and show what the resulting output XML would look like.

Note: There are many valid XML components in a Galaxy XML file. To learn more about each individual tool component, please read the Galaxy documentation.

Initialize an XML

- Click the **Create Tool XML** tab
- Enter `seqtk_seq_2.xml` into **XML file name**
- Leave **Tool description** blank for the tutorial
- Select **From scratch** and click **Save**

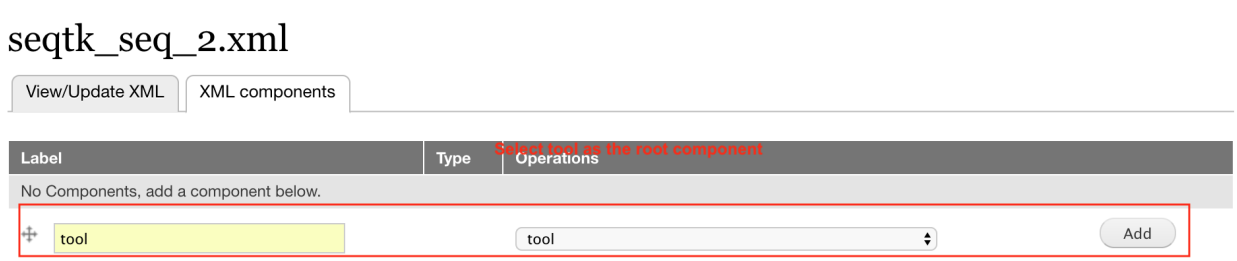


If successful, you will see the message: "The new webform `seqtk_seq_2.xml` has been created. Add new fields to your webform with the form below."

Build The Tool Components

After you create the XML file, the XML interface will be open. To reach it again, click the **Build Tool Repository** tab, and click **edit** for your tool.

1. Create the root tool component

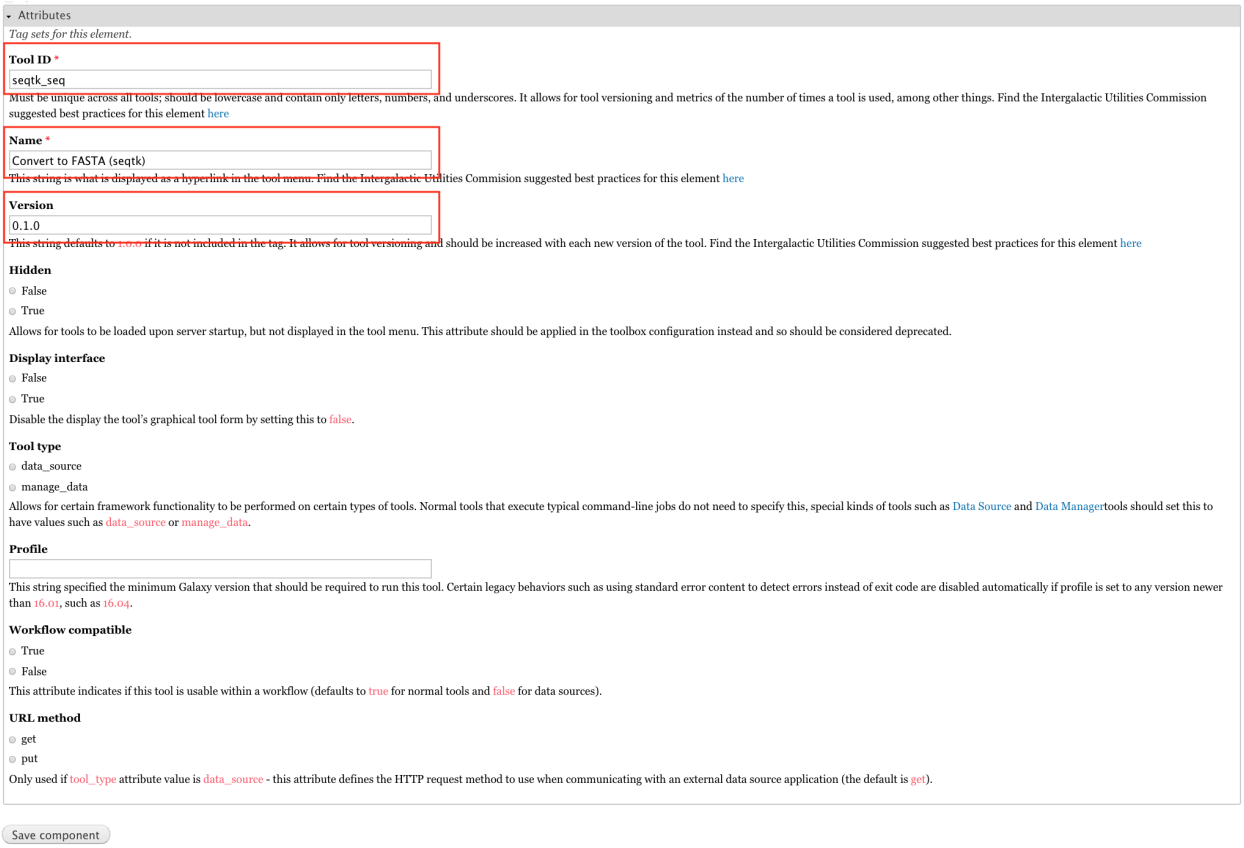


Fill out the following values for the tool root:

Table 1: root tool attributes

Field Label	Value
Tool ID	seqtk_seq
Name	Convert to FASTA (seqtk)
Version	0.1.0

Leave the other fields blank, and click **Save**.



The resulting XML element looks like this:

```
<tool id="seqtk_seq" name="Convert to FASTA (seqtk)" version="0.1.0">
```

2. Define the tool's requirements

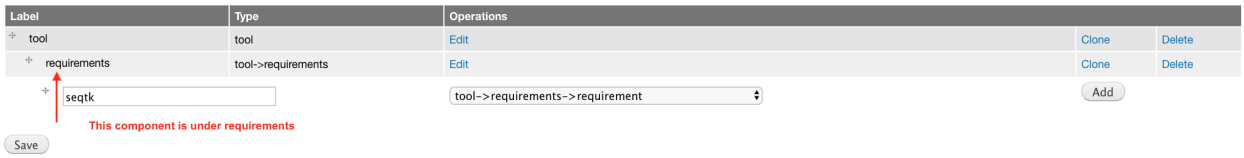
Add **tool->requirements** component

The component `tool->requirements` is a subcomponent of the component `tool`, it needs to be placed under `tool`. You can drag a component to arrange its location. **All subcomponents needs to be correctly placed under their parent components.**

Set the label to **requirements** and choose **tool->requirements** from the select box under **Operations**.

This component does not have any attributes, so just click **Save Component**. This is because the requirements parent is just a list individual requirements: let's define one next.

Next we'll build our actual requirement component. Name it `seqtk`, and select **tool->requirements->requirement** for the **Operation**.

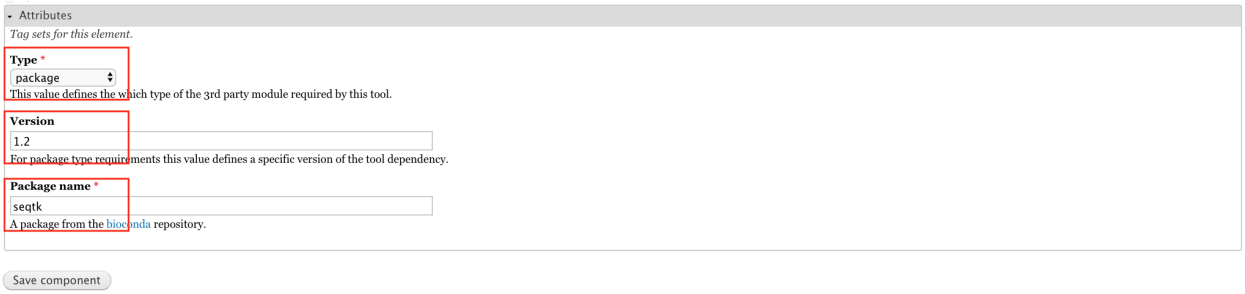


Fill out the following values for the requirements attribute:

Table 2: Requirement Attributes

Field Label	Value
Type	package
Version	1.2
Package name	seqtk

Edit **tool->requirements->requirement** component attributes.



We've just added the below XML to our tool.

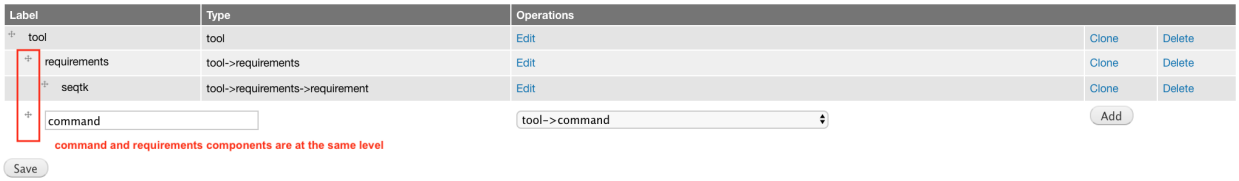
```
<requirements>
        <requirement type="package" version="1.2">seqtk</requirement>
</requirements>
```

3. Create tool->command component

Next, we will add the below XML block.

```
<command detect_errors="exit_code"><![CDATA[
    seqtk seq -a '$input1' > '$output1'
]]></command>
```

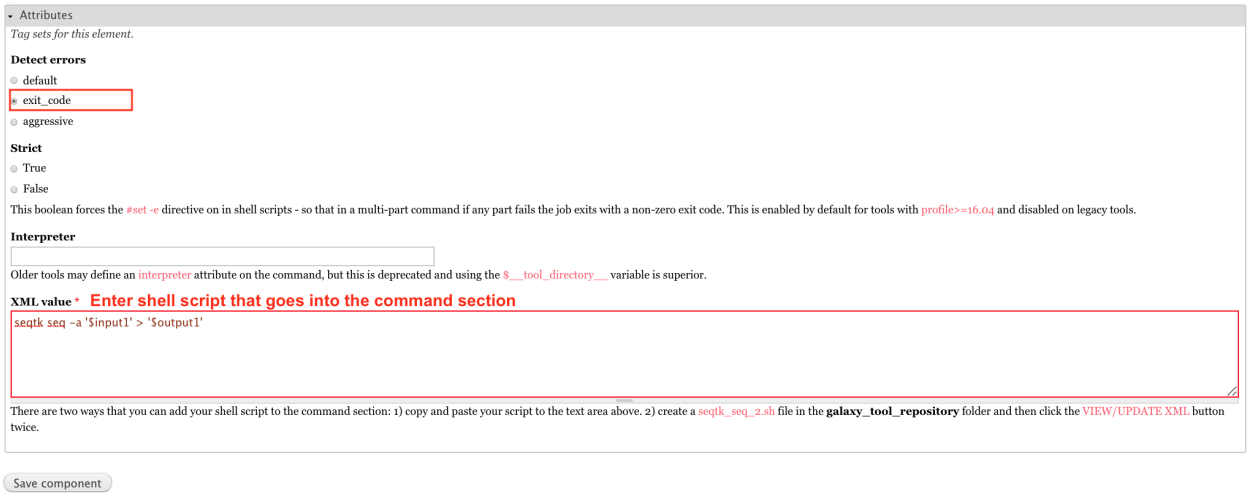
Add a component labeled **command** and select **tool->command** for the type.



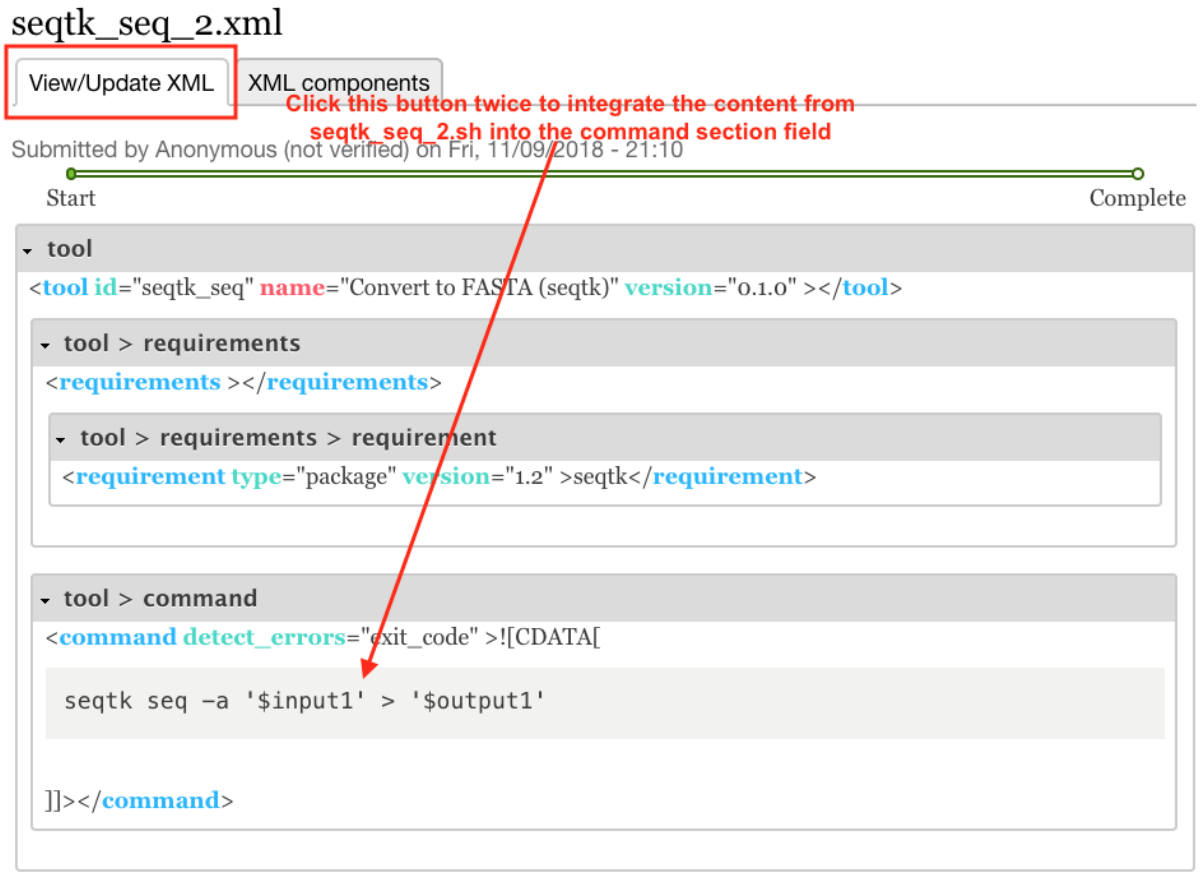
Enter the below attributes for this component:

Table 3: Command Attributes

Field Label	Value
Detect errors	exit_code
XML value	`seqtk seq -a '$input1 > $output1'`



The **XML value** field in the above web form is used to collect the shell script for the command section. However, there is an easier way to input a shell script into the tool XML file. Go to the `gtg_dev_dir/galaxy_tool_repository` and create a `.sh` file. Put the shell script into this file, and the contents will be automatically integrated into the web form field when the XML webform page is being viewed (see the image below). The `.sh` file should have exactly the same base name as the XML file. In this example, the XML file is `seqtk_seq_2.xml`, so the `.sh` file should be `seqtk_seq_2.sh`.

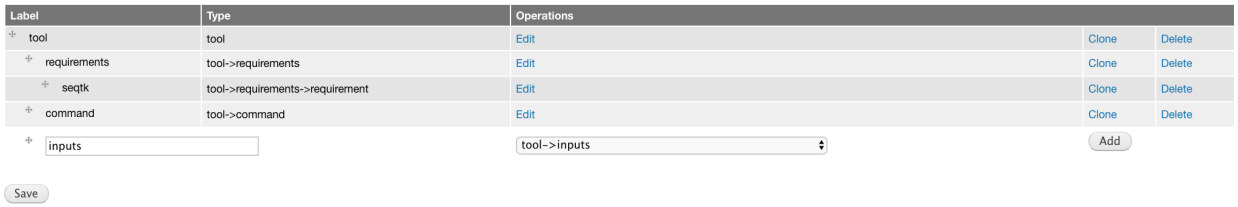


4. Create tool->inputs component

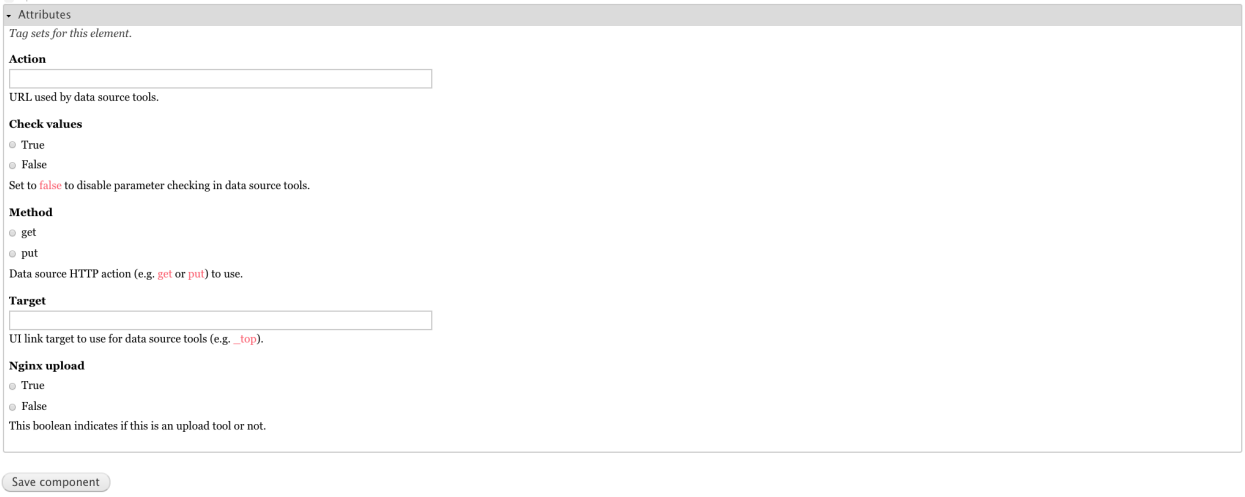
Net, we will add inputs, resulting in the following XML.

```
  <inputs>
    <param type="data" name="input1" format="fastq" />
</inputs>
```

Create a component labeled **inputs**, choosing the **tool->inputs** type.



In this example, we don't need to edit any attributes for this component, so submit the attributes form blank.



Next, add a component labeled input_data, selecting the **tool->inputs->param(type: data)** component type.

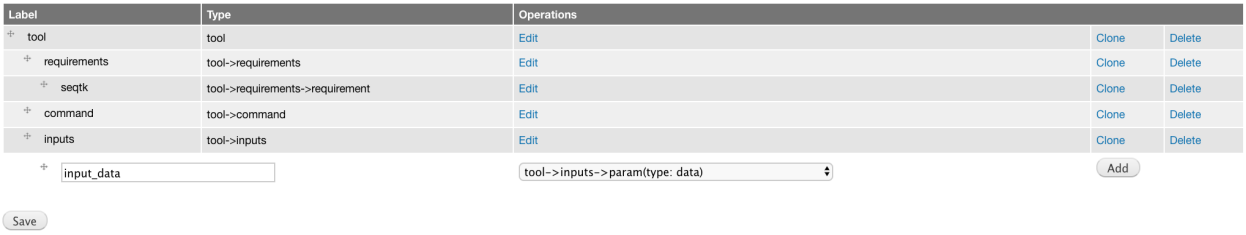


Table 4: Parameter Type Attributes

Field Label	Value
Name	input1
Format	fasta

Attributes

Tag sets for this element.

Name *

input1

Name for this element. This name is used as the Cheetah variable containing the user-supplied parameter name in command and configfile elements. The name should not contain pipes or periods (e.g. .). Some "reserved" names are REDIRECT_URL, DATA_URL, GALAXY_URL.

Argument

If the parameter reflects just one command line argument of a certain tool, this tag should be set to that particular argument. It is rendered in parenthesis after the help section, and it will create the name attribute from the argument attribute by stripping the dashes (e.g. if argument="--sensitive" then name="sensitive" is implicit).

Label

The attribute value will be displayed on the tool page as the label of the form field (label="Sort Query").

Help

Short bit of text, rendered on the tool form just below the associated field to provide information about the field. Find the Intergalactic Utilities Commision suggested best practices for this element here.

Optional

- False
- True

If false, parameter must have a value. Defaults to "false".

Refresh on change

- False
- True

Force a reload of the tool panel when the value of this parameter changes to allow code file processing. See deprecation-like notice for code blocks.

Format

fasta

Only if type attribute value is data or data_collection - the list of supported data formats is contained in the /config/datatypes_conf.xml.sample file. Use the file extension.

Multiple

- True
- False

Allow multiple valus to be selected. Valid with data and select parameters.

Save component

5. Create tool->outputs component

Next, we'll add the below XML.

```
<outputs>
    <data name="output1" format="fasta" />
</outputs>
```

Add a component labeled outputs, of type **tool->outputs**.

Label	Type	Operations		
tool	tool	Edit	Clone	Delete
requirements	tool->requirements	Edit	Clone	Delete
seqtk	tool->requirements->requirement	Edit	Clone	Delete
command	tool->command	Edit	Clone	Delete
inputs	tool->inputs	Edit	Clone	Delete
input_data	tool->inputs->param(type: data)	Edit	Clone	Delete
outputs		tool->outputs	Add	

Save

Leave the attributes blank for this component.

Attributes

Tag sets for this element.

Provided metadata style

Style used for tool provided metadata file (i.e. galaxy.json) - this can be either "legacy" or "default". The default of tools with a profile of 17.09 or newer are "default", and "legacy" for older and tools and tools without a specified profile. A discussion of the differences between the styles can be found at https://github.com/galaxyproject/galaxy/pull/4437.

Provided metadata file

Path relative to tool's working directory to load tool provided metadata from. This metadata can describe dynamic datasets to load, dynamic collection contents, as well as simple metadata (e.g. name, dbkey, etc...) and datatype specific metadata for declared outputs. More information can be found here. The default is galaxy.json.

Save component

6. Create tool->tests component

Next well create a tests component, which looks like this in XML:

```
  <tests>
    <test>
        <param name="input1" value="2.fastq"/>
        <output name="output1" file="2.fasta"/>
    </test>
</tests>
```

Add a tests component of the **tool->tests** component type.

Label	Type	Operations		
tool	tool	Edit	Clone	Delete
requirements	tool->requirements	Edit	Clone	Delete
seqtk	tool->requirements->requirement	Edit	Clone	Delete
command	tool->command	Edit	Clone	Delete
inputs	tool->inputs	Edit	Clone	Delete
input_data	tool->inputs->param(type: data)	Edit	Clone	Delete
outputs	tool->outputs	Edit	Clone	Delete
tests		tool->tests	Add	

Save

There are no attributes to choose.

Attributes

Tag sets for this element.

Save component

Add a test component of the **tool->tests->test** component type

Label	Type	Operations		
tool	tool	Edit	Clone	Delete
requirements	tool->requirements	Edit	Clone	Delete
seqtk	tool->requirements->requirement	Edit	Clone	Delete
command	tool->command	Edit	Clone	Delete
inputs	tool->inputs	Edit	Clone	Delete
input_data	tool->inputs->param(type: data)	Edit	Clone	Delete
outputs	tool->outputs	Edit	Clone	Delete
tests	tool->tests	Edit	Clone	Delete
test		tool->tests->test	Add	

Save

Again, there are no attributes to choose.

Attributes

Tag sets for this element.

Save component

Add a **tool->tests->test->param** component labeled input1.

Label	Type	Operations		
tool	tool	Edit	Clone	Delete
requirements	tool->requirements	Edit	Clone	Delete
seqtk	tool->requirements->requirement	Edit	Clone	Delete
command	tool->command	Edit	Clone	Delete
inputs	tool->inputs	Edit	Clone	Delete
input_data	tool->inputs->param(type: data)	Edit	Clone	Delete
outputs	tool->outputs	Edit	Clone	Delete
tests	tool->tests	Edit	Clone	Delete
test	tool->tests->test	Edit	Clone	Delete
input1		tool->tests->test->param	Add	

Save

For the attributes, set **Name** to `2.fastq`.

Attributes

Tag sets for this element.

Name *

2.fastq

This value must match the name of the associated input parameter (param).

Value

File type

This attribute name should be included only with parameters of type data for the tool. If this attribute name is not included, the functional test framework will attempt to determine the data type for the input dataset using the data type sniffers.

dbkey

Specifies a dbkey value for the referenced input dataset. This is only valid if the corresponding parameter is of type data.

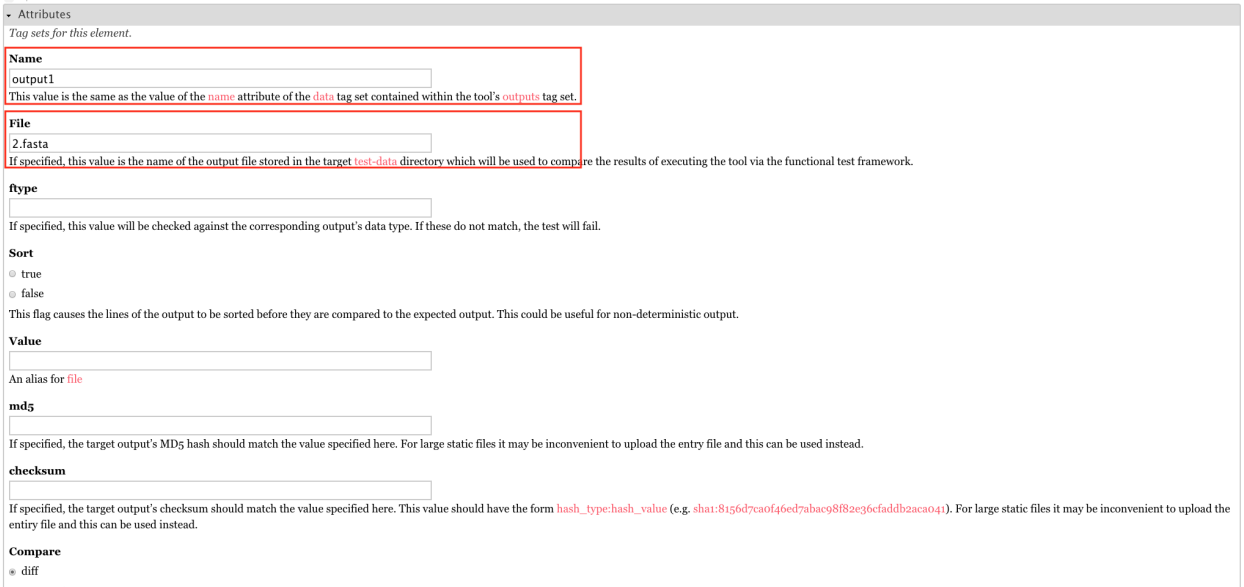
Save component

Add a **tool->tests->test-output** component labeled output1.

Label	Type	Operations		
tool	tool	Edit	Clone	Delete
requirements	tool->requirements	Edit	Clone	Delete
seqtk	tool->requirements->requirement	Edit	Clone	Delete
command	tool->command	Edit	Clone	Delete
inputs	tool->inputs	Edit	Clone	Delete
input_data	tool->inputs->param(type: data)	Edit	Clone	Delete
outputs	tool->outputs	Edit	Clone	Delete
tests	tool->tests	Edit	Clone	Delete
test	tool->tests->test	Edit	Clone	Delete
input1	tool->tests->test->param	Edit	Clone	Delete
output1		tool->tests->test->output	Add	

Save

For the attributes, set **Name** to output1 and **File** to 2.fasta



7. Create tool->help component

Next we'll provide a help component, which looks like this:

```
<help><![CDATA[

Usage:   seqtk seq [options] <in.fq>|<in.fa>
Options: -q INT    mask bases with quality lower than INT [0]
         -X INT    mask bases with quality higher than INT [255]
         -n CHAR   masked bases converted to CHAR; 0 for lowercase [0]
         -l INT    number of residues per line; 0 for 2~32-1 [0]
         -Q INT    quality shift: ASCII-INT gives base quality [33]
         -s INT    random seed (effective with -f) [11]
         -f FLOAT  sample FLOAT fraction of sequences [1]
         -M FILE   mask regions in BED or name list FILE [null]
         -L INT    drop sequences with length shorter than INT [0]
         -c        mask complement region (effective with -M)
         -r        reverse complement
         -A        force FASTA output (discard quality)
         -C        drop comments at the header lines
         -N        drop sequences containing ambiguous bases
         -1        output the 2n-1 reads only
         -2        output the 2n reads only
         -V        shift quality by '(-Q) - 33'
         -U        convert all bases to uppercases
         -S        strip of white spaces in sequences
    ]]></help>
```

Add **tool->help** component labeled help.

Label	Type	Operations		
tool	tool	Edit	Clone	Delete
requirements	tool->requirements	Edit	Clone	Delete
seqtk	tool->requirements->requirement	Edit	Clone	Delete
command	tool->command	Edit	Clone	Delete
inputs	tool->inputs	Edit	Clone	Delete
input_data	tool->inputs->param(type: data)	Edit	Clone	Delete
outputs	tool->outputs	Edit	Clone	Delete
tests	tool->tests	Edit	Clone	Delete
test	tool->tests->test	Edit	Clone	Delete
input1	tool->tests->test->param	Edit	Clone	Delete
output1	tool->tests->test->output	Edit	Clone	Delete
help		tool->help	Add	

Save

For the attributes, paste the below text into the **XML value** field.

```
Usage:   seqtk seq [options] <in.fq>|<in.fa>
Options: -q INT    mask bases with quality lower than INT [0]
         -X INT    mask bases with quality higher than INT [255]
         -n CHAR   masked bases converted to CHAR; 0 for lowercase [0]
         -l INT    number of residues per line; 0 for 2~32-1 [0]
         -Q INT    quality shift: ASCII-INT gives base quality [33]
         -s INT    random seed (effective with -f) [11]
         -f FLOAT  sample FLOAT fraction of sequences [1]
         -M FILE   mask regions in BED or name list FILE [null]
         -L INT    drop sequences with length shorter than INT [0]
         -c        mask complement region (effective with -M)
         -r        reverse complement
         -A        force FASTA output (discard quality)
         -C        drop comments at the header lines
         -N        drop sequences containing ambiguous bases
         -1        output the 2n-1 reads only
         -2        output the 2n reads only
         -V        shift quality by '(-Q) - 33'
         -U        convert all bases to uppercases
         -S        strip of white spaces in sequences
```

Attributes

Tag sets for this element.

XML value *

```
Usage:   seqtk seq [options] <in.fq>|<in.fa>
Options: -q INT    mask bases with quality lower than INT [0]
         -X INT    mask bases with quality higher than INT [255]
         -n CHAR   masked bases converted to CHAR; 0 for lowercase [0]
         -l INT    number of residues per line; 0 for 2^32-1 [0]
         -Q INT    quality shift: ASCII-INT gives base quality [33]
         -s INT    random seed (effective with -f) [11]
         -f FLOAT  sample FLOAT fraction of sequences [1]
         -M FILE   mask regions in BED or name list FILE [null]
         -L INT    drop sequences with length shorter than INT [0]
         -c        mask complement region (effective with -M)
         -r        reverse complement
         -A        force FASTA output (discard quality)
         -C        drop comments at the header lines
         -N        drop sequences containing ambiguous bases
         -1        output the 2n-1 reads only
         -2        output the 2n reads only
         -V        shift quality by '(-Q) - 33'
         -U        convert all bases to uppercases
         -S        strip of white spaces in sequences
```

This tag set includes all of the necessary details of how to use the tool. This tag set should be included as the next to the last tag set, before citations, in the tool config. Tool help is written in reStructuredText. Included here is only an overview of a subset of features. For more information see http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html.

Save component

8. Create tool->citations component

Finally, we will create a citation component.

```
<citations>
        <citation type="bibtex">
@misc{githubseqtk,
  author = {LastTODO, FirstTODO},
  year = {TODO},
  title = {seqtk},
  publisher = {GitHub},
  journal = {GitHub repository},
  url = {https://github.com/lh3/seqtk},
}</citation>
    </citations>
```

Add **tool->citations** component labeled citations.

Label	Type	Operations		
tool	tool	Edit	Clone	Delete
requirements	tool->requirements	Edit	Clone	Delete
seqtk	tool->requirements->requirement	Edit	Clone	Delete
command	tool->command	Edit	Clone	Delete
inputs	tool->inputs	Edit	Clone	Delete
input_data	tool->inputs->param(type: data)	Edit	Clone	Delete
outputs	tool->outputs	Edit	Clone	Delete
tests	tool->tests	Edit	Clone	Delete
test	tool->tests->test	Edit	Clone	Delete
input1	tool->tests->test->param	Edit	Clone	Delete
output1	tool->tests->test->output	Edit	Clone	Delete
help	tool->help	Edit	Clone	Delete
citations	tool->citations		Add	

Save

This component does not have attributes.

Attributes

Tag sets for this element.

Save component

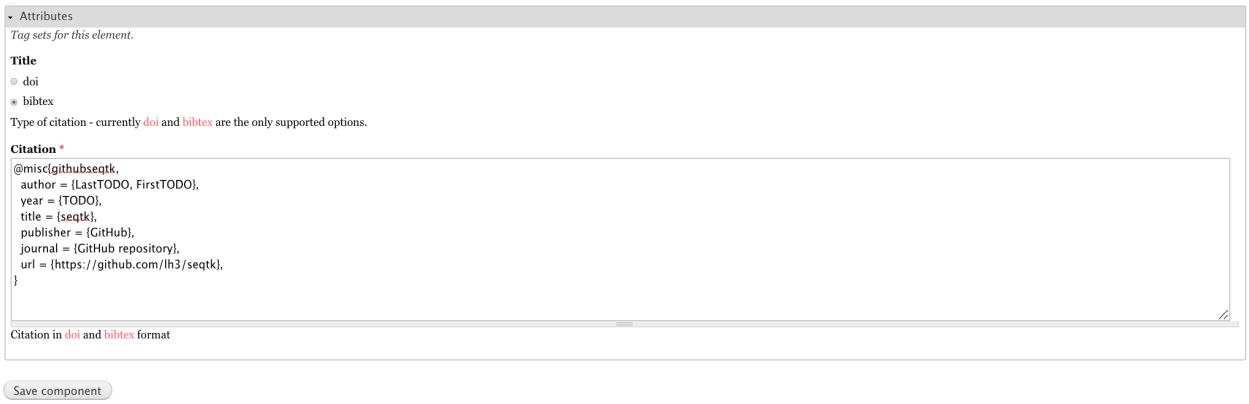
Add **tool->citations->citation** component labeled citation githubseqtk.

Label	Type	Operations		
tool	tool	Edit	Clone	Delete
requirements	tool->requirements	Edit	Clone	Delete
seqtk	tool->requirements->requirement	Edit	Clone	Delete
command	tool->command	Edit	Clone	Delete
inputs	tool->inputs	Edit	Clone	Delete
input_data	tool->inputs->param(type: data)	Edit	Clone	Delete
outputs	tool->outputs	Edit	Clone	Delete
tests	tool->tests	Edit	Clone	Delete
test	tool->tests->test	Edit	Clone	Delete
input1	tool->tests->test->param	Edit	Clone	Delete
output1	tool->tests->test->output	Edit	Clone	Delete
help	tool->help	Edit	Clone	Delete
citations	tool->citations	Edit	Clone	Delete
citation githubseqtk	tool->citations->citation		Add	

Save

For the attributes, select bibtex for the **Title**, and paste the below citation in the **Citation** field.

```
@misc{githubseqtk,
  author = {LastTODO, FirstTODO},
  year = {TODO},
  title = {seqtk},
  publisher = {GitHub},
  journal = {GitHub repository},
  url = {https://github.com/lh3/seqtk},
}
```

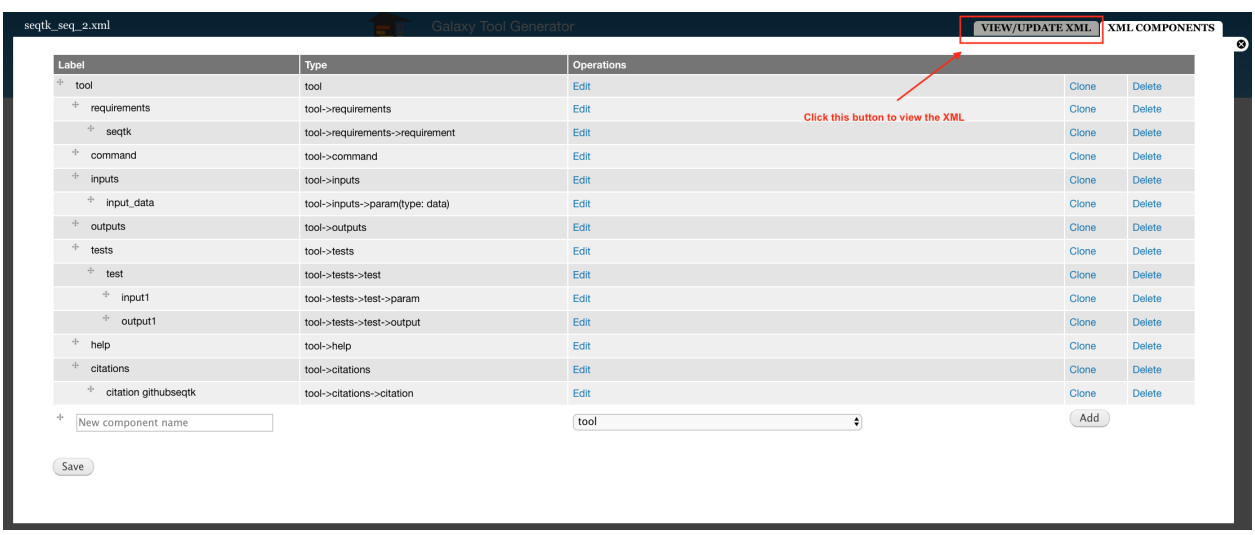


View the complete XML file

Now you have created all the components for building the `seqtk_seq_2.xml` file, you can view the XML page to see how it looks on GTG. Of course, you can view the XML page any time you want. It doesn't have to be after you have added all the components.

To view the built XML, click the **VIEW/UPDATE XML** tab from the edit component page.

Note: You can also view the final XML from the **Build Tools Repository** page by clicking the **view** button.



Below is the XML page.

seqtk_seq_2.xml

View/Update XML | XML components

Submitted by Anonymous (not verified) on Fri, 11/09/2018 - 21:10

Start — Complete

tool

```
<tool id="seqtk_seq" name="Convert to FASTA (seqtk)" version="0.1.0" ></tool>
```

tool > requirements

```
<requirements ></requirements>
```

tool > requirements > requirement

```
<requirement type="package" version="1.2" >seqtk</requirement>
```

tool > command

```
<command detect_errors="exit_code" >![CDATA[

seqtk seq -a '$input1' > '$output1'

]]></command>
```

tool > inputs

```
<inputs ></inputs>
```

tool > inputs > param (type: data)

```
<param type="data" name="input1" optional="False" format="fasta" ></param>
```

tool > outputs

```
<outputs ></outputs>
```

tool > tests

```
<tests ></tests>
```

tool > tests > test

```
<test ></test>
```

tool > tests > test > param

```
<param name="2.fastq" ></param>
```

tool > tests > test > output

```
<output name="output1" file="2.fasta" compare="diff" ></output>
```

tool > help

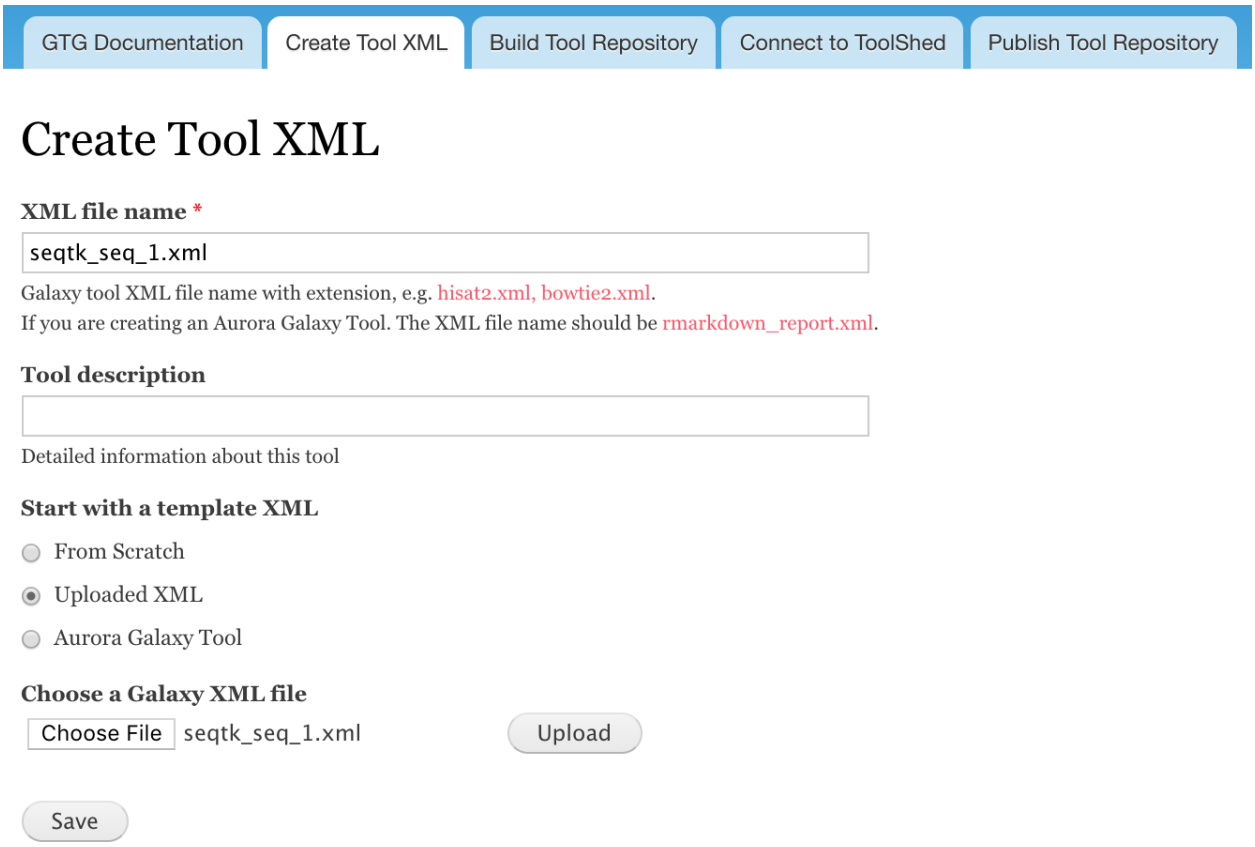
```
<help >![CDATA[

Usage:   seqtk seq [options] |
```

3.2.2 Uploaded XML

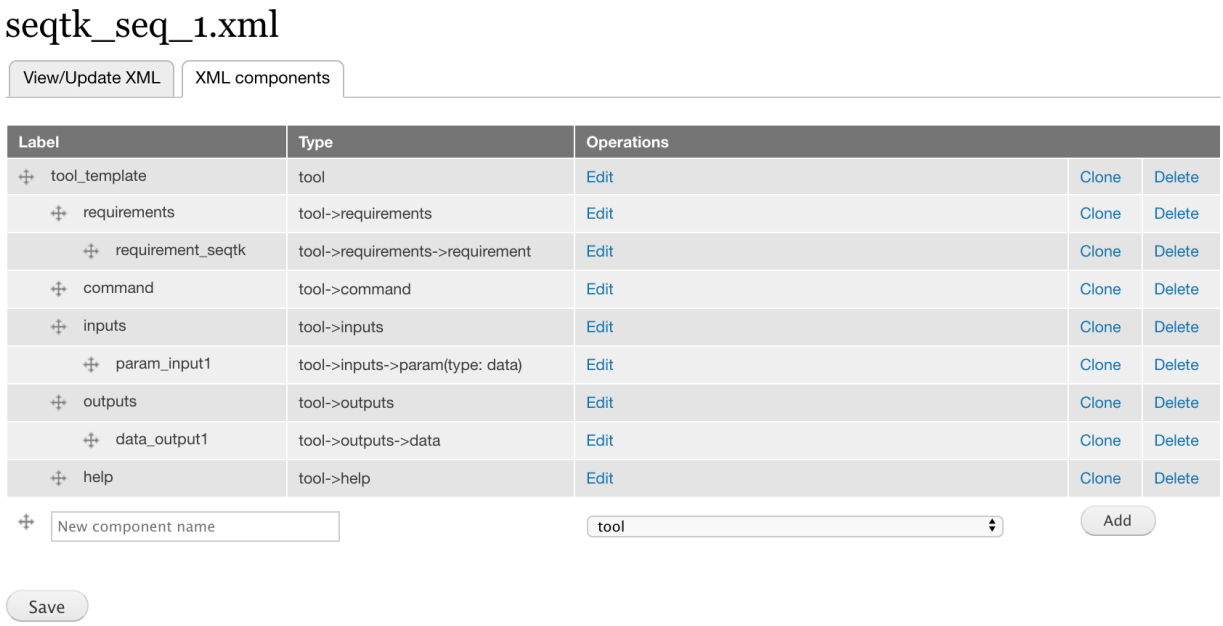
GTG allows uploading an existing XML file and building web components upon it. In this section, we will show how to build `seqtk_seq_2.xml` from `seqtk_seq_1.xml`.

Upload XML

- Click the **Create Tool XML** tab
- Enter `seqtk_seq_2.xml` into **XML file name**
- Leave **Tool description** blank for the tutorial
- Select **Uploaded XML**
- Click **Choose File** and select *seqtk_seq_1.xml* in your computer and click **Upload**
- Click **Save**



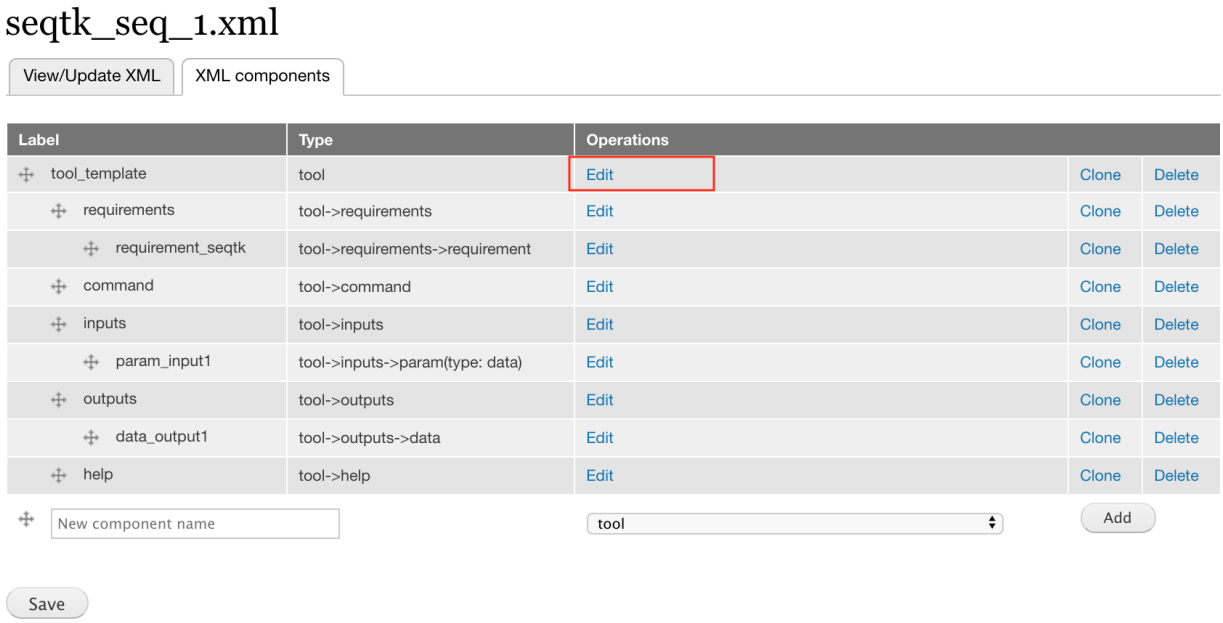
You should be redirected to the webform components page. If not, you can click the **Build Tool Repository** table, and click **edit** for the XML you just created.



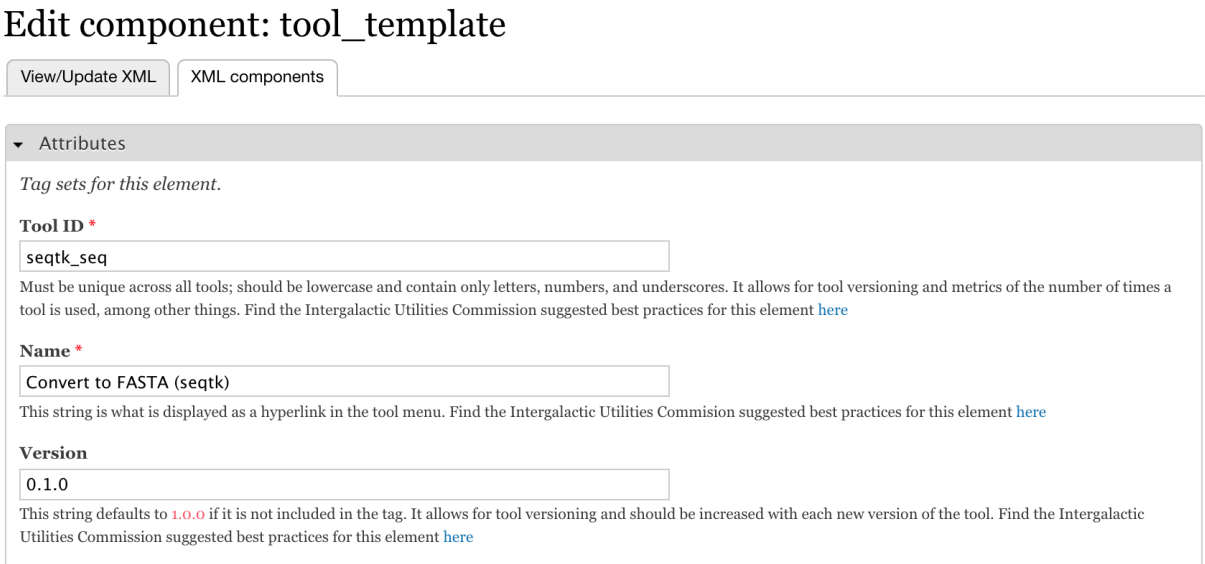
Correct Tool ID attribute

When you upload an XML file, the **Tool ID** attribute in the **tool** component is always `tool_1`. We need to correct this attribute.

- Click **edit** for the **tool** component on the component page.



- **This will open the edit form for the tool component, through which you can edit the attributes.**
  - Replace `tool_1` with `seqtk_seq`.
  - Click **Save component**



Add more components

Compared to the *seqtk_seq_2.xml*, *seqtk_seq_1.xml* is missing the following components. We are going to add them one by one.

The tool->tests component

```
<tests>
    <test>
        <param name="input1" value="2.fastq"/>
        <output name="output1" file="2.fasta"/>
    </test>
</tests>
```

Add a tests component of the **tool->tests** component type and drag it to the correct location.

The component `tool->test` is a subcomponent of the component `tool`. It needs to be placed under `tool` and at the same level as other components like `tool->requirements`, `tool->command`, `tool->inputs`, `tool->outputs`, and `tool->help`. You can drag a component to arrange its location. **All subcomponents needs to be correctly placed under their parent components.**

Label	Type	Operations		
tool_template	tool	Edit	Clone	Delete
requirements	tool->requirements	Edit	Clone	Delete
requirement_seqtk	tool->requirements->requirement	Edit	Clone	Delete
command	tool->command	Edit	Clone	Delete
inputs	tool->inputs	Edit	Clone	Delete
param_input1	tool->inputs->param(type: data)	Edit	Clone	Delete
outputs	tool->outputs	Edit	Clone	Delete
data_output1	tool->outputs->data	Edit	Clone	Delete
tests		tool->tests	Add	
help	tool->help	Edit	Clone	Delete

Save

There are no attributes to choose.

Attributes

Tag sets for this element.

Save component

Add a test component of the **tool->tests->test** component type and place it under the tests component.

Label	Type	Operations		
tool_template	tool	Edit	Clone	Delete
requirements	tool->requirements	Edit	Clone	Delete
requirement_seqtk	tool->requirements->requirement	Edit	Clone	Delete
command	tool->command	Edit	Clone	Delete
inputs	tool->inputs	Edit	Clone	Delete
param_input1	tool->inputs->param(type: data)	Edit	Clone	Delete
tests	tool->tests	Edit	Clone	Delete
test		tool->tests->test	Add	
outputs	tool->outputs	Edit	Clone	Delete
data_output1	tool->outputs->data	Edit	Clone	Delete
help	tool->help	Edit	Clone	Delete

Save

Again, there are no attributes to choose.

Attributes

Tag sets for this element.

Save component

Add a **tool->tests->test->param** component labeled input1.

Label	Type	Operations		
tool_template	tool	Edit	Clone	Delete
requirements	tool->requirements	Edit	Clone	Delete
requirement_seqtk	tool->requirements->requirement	Edit	Clone	Delete
command	tool->command	Edit	Clone	Delete
inputs	tool->inputs	Edit	Clone	Delete
param_input1	tool->inputs->param(type: data)	Edit	Clone	Delete
tests	tool->tests	Edit	Clone	Delete
test	tool->tests->test	Edit	Clone	Delete
input1		tool->tests->test->param	Add	
outputs	tool->outputs	Edit	Clone	Delete
data_output1	tool->outputs->data	Edit	Clone	Delete
help	tool->help	Edit	Clone	Delete

Save

For the attributes, set **Name** to `2.fastq`.

Attributes

Tag sets for this element.

Name *

2.fastq

This value must match the name of the associated input parameter (param).

Value

File type

This attribute name should be included only with parameters of type data for the tool. If this attribute name is not included, the functional test framework will attempt to determine the data type for the input dataset using the data type sniffers.

dbkey

Specifies a dbkey value for the referenced input dataset. This is only valid if the corresponding parameter is of type data.

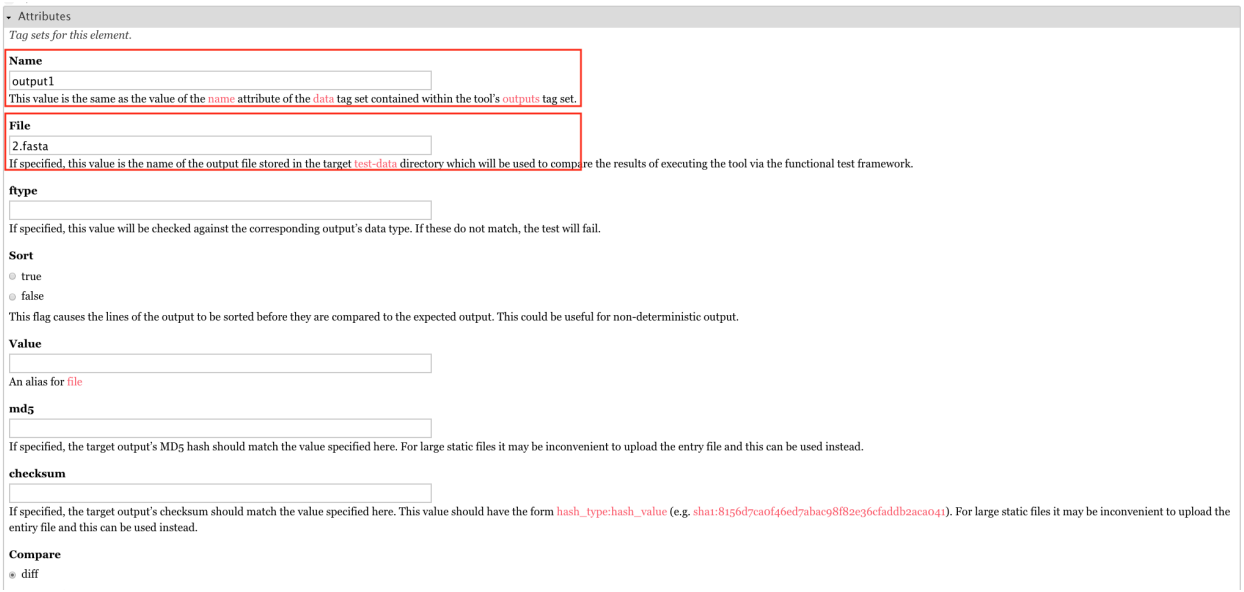
Save component

Add a **tool->tests->test-output** component labeled output1.

Label	Type	Operations		
tool_template	tool	Edit	Clone	Delete
requirements	tool->requirements	Edit	Clone	Delete
requirement_seqtk	tool->requirements->requirement	Edit	Clone	Delete
command	tool->command	Edit	Clone	Delete
inputs	tool->inputs	Edit	Clone	Delete
param_input1	tool->inputs->param(type: data)	Edit	Clone	Delete
tests	tool->tests	Edit	Clone	Delete
test	tool->tests->test	Edit	Clone	Delete
input1	tool->tests->test->param	Edit	Clone	Delete
output1		tool->tests->test->output	Add	
outputs	tool->outputs	Edit	Clone	Delete
data_output1	tool->outputs->data	Edit	Clone	Delete
help	tool->help	Edit	Clone	Delete

Save

For the attributes, set **Name** to output1 and **File** to 2.fasta



The content in the tool->help component

```
Usage:   seqtk seq [options] <in.fq>|<in.fa>

Options: -q INT    mask bases with quality lower than INT [0]
         -X INT    mask bases with quality higher than INT [255]
         -n CHAR   masked bases converted to CHAR; 0 for lowercase [0]
         -l INT    number of residues per line; 0 for 2^32-1 [0]
         -Q INT    quality shift: ASCII-INT gives base quality [33]
         -s INT    random seed (effective with -f) [11]
         -f FLOAT  sample FLOAT fraction of sequences [1]
         -M FILE   mask regions in BED or name list FILE [null]
         -L INT    drop sequences with length shorter than INT [0]
         -c        mask complement region (effective with -M)
         -r        reverse complement
         -A        force FASTA output (discard quality)
         -C        drop comments at the header lines
         -N        drop sequences containing ambiguous bases
         -1        output the 2n-1 reads only
         -2        output the 2n reads only
         -V        shift quality by '(-Q) - 33'
         -U        convert all bases to uppercases
         -S        strip of white spaces in sequences
```

The uploaded XML already has a **tool->help** component. We just need to open the component edit form and fill in the content above.

Label	Type	Operations		
tool_template	tool	Edit	Clone	Delete
requirements	tool->requirements	Edit	Clone	Delete
requirement_seqtk	tool->requirements->requirement	Edit	Clone	Delete
command	tool->command	Edit	Clone	Delete
inputs	tool->inputs	Edit	Clone	Delete
param_input1	tool->inputs->param(type: data)	Edit	Clone	Delete
tests	tool->tests	Edit	Clone	Delete
test	tool->tests->test	Edit	Clone	Delete
input1	tool->tests->test->param	Edit	Clone	Delete
output1	tool->tests->test->output	Edit	Clone	Delete
outputs	tool->outputs	Edit	Clone	Delete
data_output1	tool->outputs->data	Edit	Clone	Delete
help	tool->help	Edit	Clone	Delete
New component name		tool	Add	

Save

For the attributes, paste the below text into the **XML value** field.

```
Usage:   seqtk seq [options] <in.fq>|<in.fa>
Options: -q INT    mask bases with quality lower than INT [0]
         -X INT    mask bases with quality higher than INT [255]
         -n CHAR   masked bases converted to CHAR; 0 for lowercase [0]
         -l INT    number of residues per line; 0 for 2~32-1 [0]
         -Q INT    quality shift: ASCII-INT gives base quality [33]
         -s INT    random seed (effective with -f) [11]
         -f FLOAT  sample FLOAT fraction of sequences [1]
         -M FILE   mask regions in BED or name list FILE [null]
         -L INT    drop sequences with length shorter than INT [0]
         -c        mask complement region (effective with -M)
         -r        reverse complement
         -A        force FASTA output (discard quality)
         -C        drop comments at the header lines
         -N        drop sequences containing ambiguous bases
         -1        output the 2n-1 reads only
         -2        output the 2n reads only
         -V        shift quality by '(-Q) - 33'
         -U        convert all bases to uppercases
         -S        strip of white spaces in sequences
```

Attributes

Tag sets for this element.

XML value *

```
Usage:   seqtk seq [options] <in.fq>|<in.fa>
Options: -q INT    mask bases with quality lower than INT [0]
         -X INT    mask bases with quality higher than INT [255]
         -n CHAR   masked bases converted to CHAR; 0 for lowercase [0]
         -l INT    number of residues per line; 0 for 2^32-1 [0]
         -Q INT    quality shift: ASCII-INT gives base quality [33]
         -s INT    random seed (effective with -f) [11]
         -f FLOAT  sample FLOAT fraction of sequences [1]
         -M FILE   mask regions in BED or name list FILE [null]
         -L INT    drop sequences with length shorter than INT [0]
         -c        mask complement region (effective with -M)
         -r        reverse complement
         -A        force FASTA output (discard quality)
         -C        drop comments at the header lines
         -N        drop sequences containing ambiguous bases
         -1        output the 2n-1 reads only
         -2        output the 2n reads only
         -V        shift quality by '(-Q) - 33'
         -U        convert all bases to uppercases
         -S        strip of white spaces in sequences
```

This tag set includes all of the necessary details of how to use the tool. This tag set should be included as the next to the last tag set, before citations, in the tool config. Tool help is written in reStructuredText. Included here is only an overview of a subset of features. For more information see http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html.

Save component

The tool->citations component

```
<citations>
            <citation type="bibtex">
            @misc{githubseqtk,
              author = {LastTODO, FirstTODO},
              year = {TODO},
              title = {seqtk},
              publisher = {GitHub},
              journal = {GitHub repository},
              url = {https://github.com/lh3/seqtk},
            }</citation>
</citations>
```

Add **tool->citations** component labeled citations.

Label	Type	Operations		
+ tool	tool	Edit	Clone	Delete
+ requirements	tool->requirements	Edit	Clone	Delete
+ seqtk	tool->requirements->requirement	Edit	Clone	Delete
+ command	tool->command	Edit	Clone	Delete
+ inputs	tool->inputs	Edit	Clone	Delete
+ input_data	tool->inputs->param(type: data)	Edit	Clone	Delete
+ outputs	tool->outputs	Edit	Clone	Delete
+ tests	tool->tests	Edit	Clone	Delete
+ test	tool->tests->test	Edit	Clone	Delete
+ input1	tool->tests->test->param	Edit	Clone	Delete
+ output1	tool->tests->test->output	Edit	Clone	Delete
+ help	tool->help	Edit	Clone	Delete
+ citations		tool->citations	Add	

Save

This component does not have attributes.

Attributes

Tag sets for this element.

Save component

Add **tool->citations->citation** component labeled citation githubseqtk.

Label	Type	Operations		
tool	tool	Edit	Clone	Delete
requirements	tool->requirements	Edit	Clone	Delete
seqtk	tool->requirements->requirement	Edit	Clone	Delete
command	tool->command	Edit	Clone	Delete
inputs	tool->inputs	Edit	Clone	Delete
input_data	tool->inputs->param(type: data)	Edit	Clone	Delete
outputs	tool->outputs	Edit	Clone	Delete
tests	tool->tests	Edit	Clone	Delete
test	tool->tests->test	Edit	Clone	Delete
input1	tool->tests->test->param	Edit	Clone	Delete
output1	tool->tests->test->output	Edit	Clone	Delete
help	tool->help	Edit	Clone	Delete
citations	tool->citations	Edit	Clone	Delete
citation githubseqtk	tool->citations->citation		Add	

Save

For the attributes, select bibtex for the **Title**, and paste the below citation in the **Citation** field.

```
@misc{githubseqtk,
  author = {LastTODO, FirstTODO},
  year = {TODO},
  title = {seqtk},
  publisher = {GitHub},
  journal = {GitHub repository},
  url = {https://github.com/lh3/seqtk},
}
```

Attributes

Tag sets for this element.

Title

doi

bibtex

Type of citation - currently doi and bibtex are the only supported options.

Citation *

```
@misc{githubseqtk,
  author = {LastTODO, FirstTODO},
  year = {TODO},
  title = {seqtk},
  publisher = {GitHub},
  journal = {GitHub repository},
  url = {https://github.com/lh3/seqtk},
}
```

Citation in doi and bibtex format

Save component

View the complete XML file

To view the complete XML file, you can following the instruction from the **From Scratch** guide.

3.2.3 Aurora Galaxy Tool

> **Warning:** Aurora Galaxy Tools isn't published yet! The github repo is here. Follow me on twitter for updates and a guide when its out.

3.3 Final Steps and Publishing

3.3.1 Building the Finished Galaxy Tool

Now that the XML file is ready, there are some final steps for making the tool available on Galaxy ToolShed.

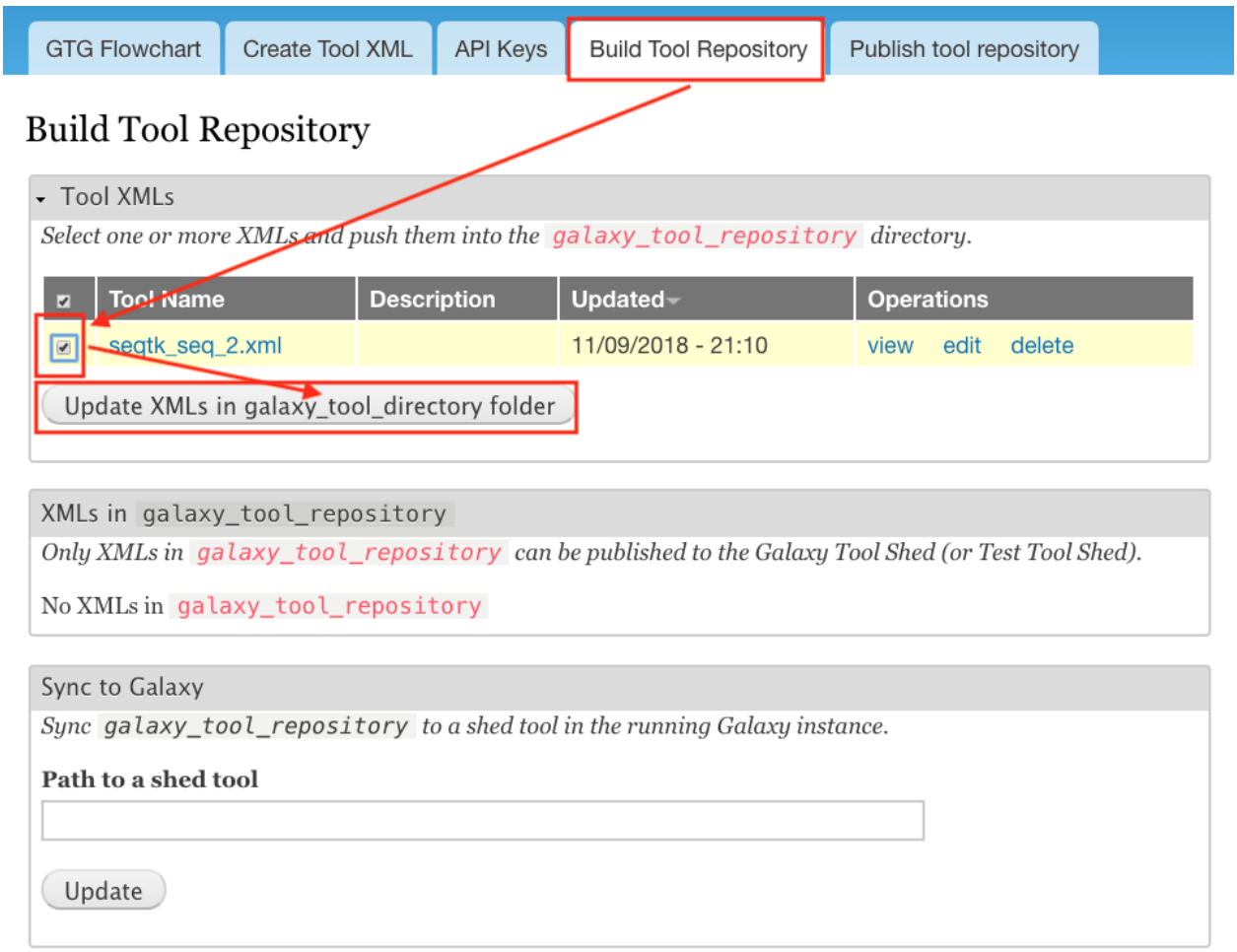
Add Files

Add XML files to the `galaxy_tool_repository` directory

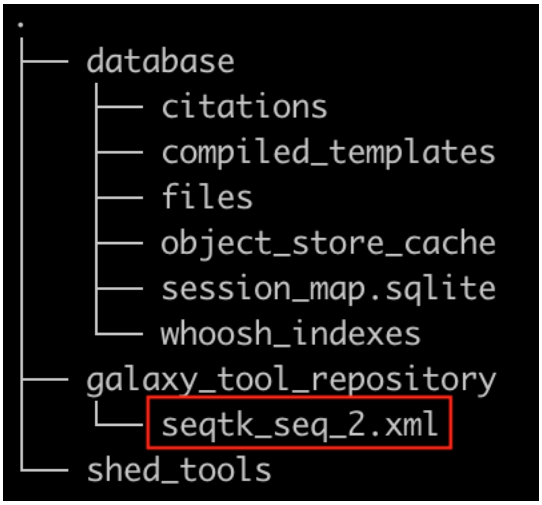
You have just created the `seqtk_seq_2.xml` file in GTG. However, this file is not in the `gtg_dev_dir/galaxy_tool_repository` directory yet. We need to copy the XML file into it, and any other non-XML files if there are any.

Click the **Build Tool Repository** tab and select any XML files that you want to add to the `gtg_dev_dir/galaxy_tool_repository` directory. And then click the **Update XMLs in galaxy_tool_directory folder** button.

Note: This is also the button that you use to add an updated XML to the directory.



You should be able to see the `seqtk_seq_2.xml` file in the `gtg_dev_dir/galaxy_tool_repository` directory.

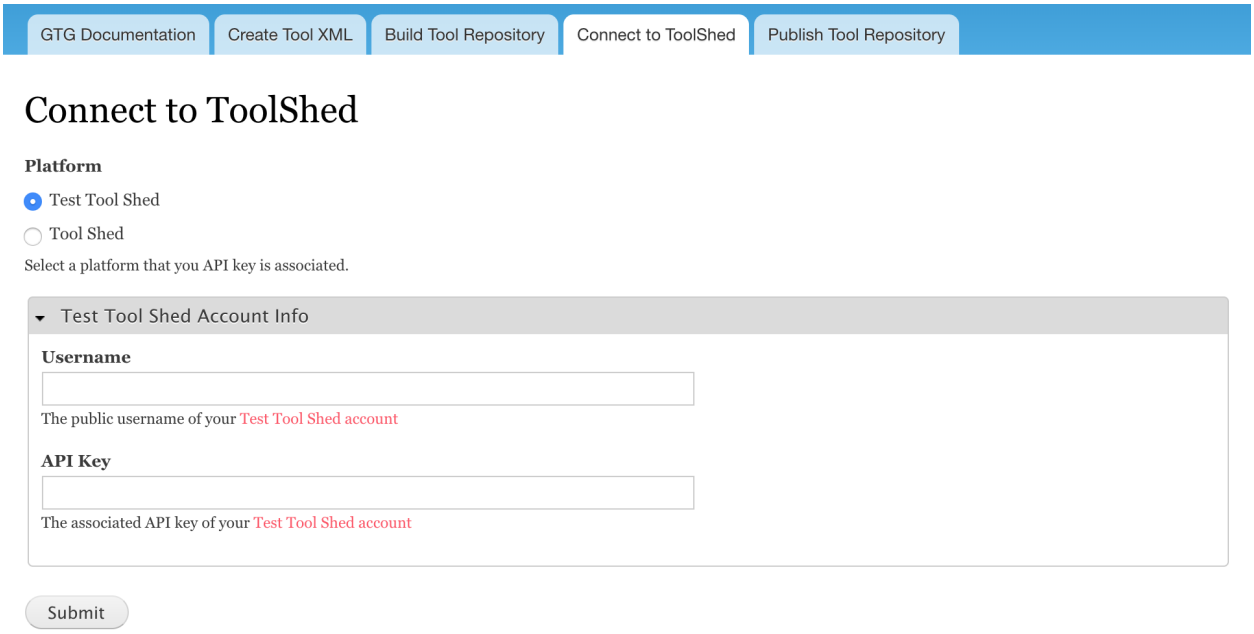


Add non-XML files to `galaxy_tool_repository`

If this tool requires any other non-XML files (for example, test files, scripts, etc.), you can add them directly to the `gtg_dev_dir/galaxy_tool_repository` directory.

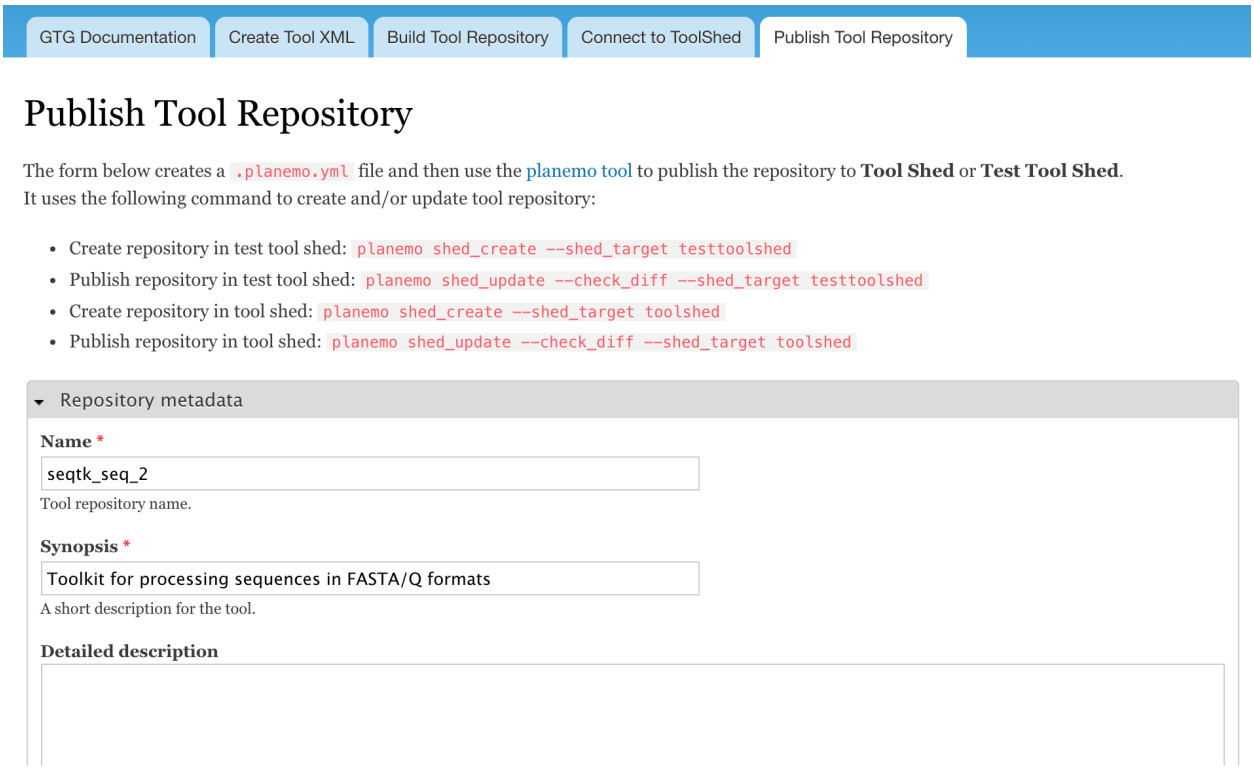
Connect to ToolShed

Once we have the XML file(s) and all other non-XML files in the `gtg_dev_dir/galaxy_tool_repository`, we can publish the tool to Test ToolShed or ToolShed with GTG. We need to connect to the Galaxy ToolShed or Test ToolShed to publish Galaxy tools. This can be down by adding the API keys through the following interface. Visit the Toolshed documentation to learn more about API keys: https://docs.galaxyproject.org/en/release_18.05/api/ts_api.html



Publish to Tool Repository

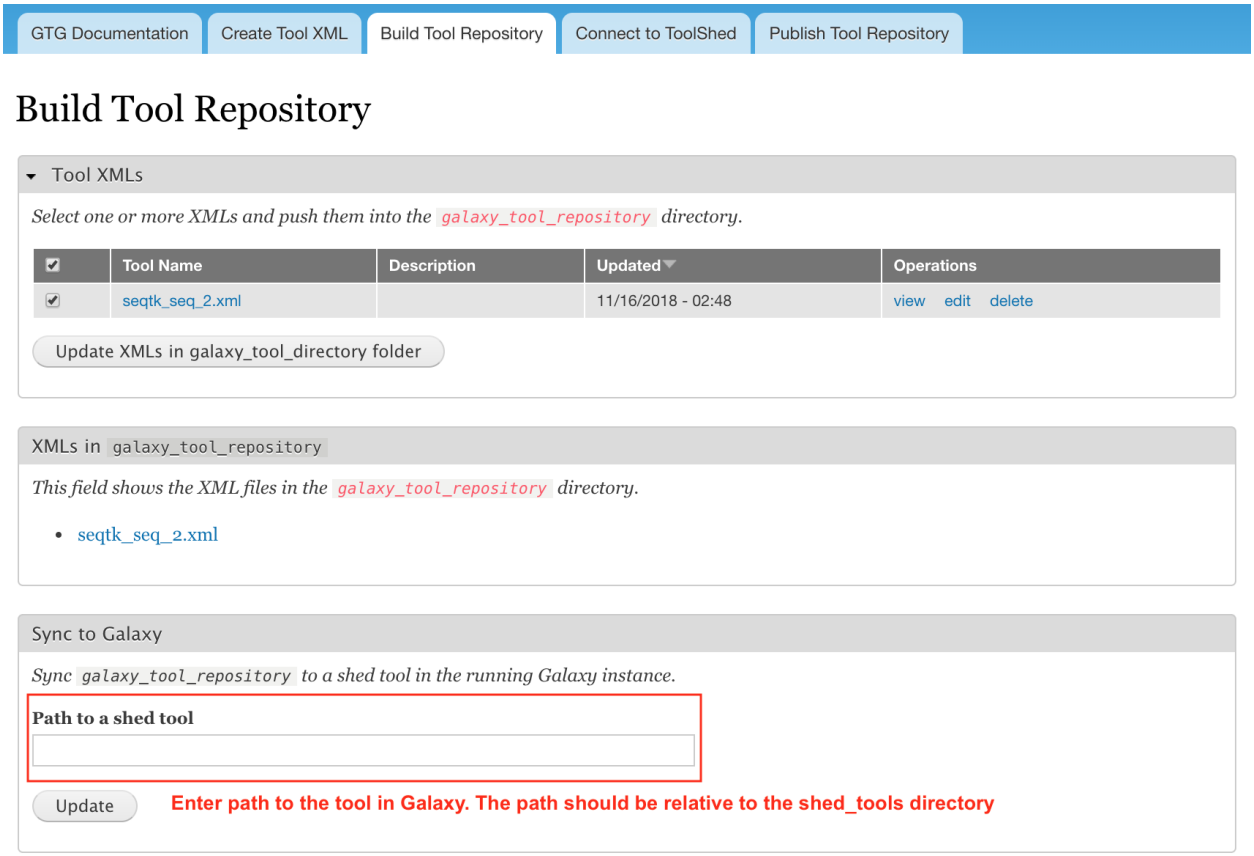
After we have connected with a ToolShed platform, we can publish the tool through the interface below.



Install and test Tool in Galaxy

The next step would be to install and test the tool in the connected Galaxy instance. If the tool needs more work, you can use GTG to update the XML file.

The **Sync to Galaxy** field on the **Build Tool Repository** page is used to link the tool in GTG with the same tool installed in Galaxy so that the update will be automatically synced to Galaxy for testing.



Every time you update the XML file in Galaxy, you will need to restart Galaxy to integrate the updates. Below is the command to restart Galaxy.

```
docker exec -it gtg_galaxy sh -c 'supervisorctl restart galaxy:'
```

You should see the following stdout.

```
galaxy:galaxy_nodejs_proxy: stopped
galaxy:handler0: stopped
galaxy:handler1: stopped
galaxy:galaxy_web: stopped
galaxy:galaxy_nodejs_proxy: started
galaxy:galaxy_web: started
galaxy:handler0: started
galaxy:handler1: started
```

CHAPTER 4

Developer Guide

Galaxy Tool Generator consists of two Drupal modules: `galaxy_tool_generator_ui` and `galaxy_tool_generator`. The `galaxy_tool_generator_ui` is responsible for the UI design of the web application. The `galaxy_tool_generator` creates a list of web form components that map to the Galaxy Tool XML components defined here. Developers can contribute to this application by creating new web form components for newly added XML components by the Galaxy project team. This guide assumes you know the basic of Drupal module development and are familiar with the Drupal Form API.

4.1 Develop Web Form Component

4.1.1 Step 0: choose a good component name

The component name should reflect the XML component structure. Below are a few examples showing the relationship between web component name and XML component:

- XML component: `tool` – webform component name: `tool`
- XML component: `tool->requirements` – webform component name: `tool_requirements`
- XML component: `tool->requirements->requirement` – webform component name: `tool->requirements->requirement`

4.1.2 Step 1: define a new webform component

Add component definition into the *hook_webform_component_info()* in the **.module** file, for example:

```
$components['tool'] = [
    'label' => 'COMPONENT_NAME',
    'features' => [
      'group' => TRUE,
    ],
```

(continues on next page)

(continued from previous page)

```
    'file' => 'components/COMPONENT_NAME.inc',
  ];
```

4.1.3 Step 2: declare a form for editing webform component attributes

Add a case entry to the *galaxy_tool_generator_form_webform_component_edit_form_alter()* in the **.module** file, for example:

```
case 'COMPONENT_NAME':
      edit_component_COMPONENT_NAME($form);
      break;
```

You will need to replace *COMPONENT_NAME* in the code block with the actual component name.

4.1.4 Step 3: define the form for editing webform component attributes

Step 3.1: utilize `component_template.inc` file

- Using the **components/component_template.inc** as a template to create component a **COMPONENT_NAME.inc**

file and place it within **./components/** folder. Replace `COMPONENT_NAME` in the file name with actual component name.

- Replace *component_template* with component name
- Fill in the *fieldset_title* argument value in the following code chunk:

```
function _webform_render_component_template($component, $value = NULL, $filter = TRUE,
↪ $submission = NULL) {
  return get_comp   onent_render_array('component_template', $component, $fieldset_
↪title = '');
}
```

Step 3.2: specify Galaxy Tool XML tag

Replace **xml_tag** in the following code chunk with actual **Galaxy Tool XML tag**:

```
/**
 * Implement edit command function.
 */
function edit_component_component_template(&$form) {
  unset($form['validation']);
  unset($form['display']);

  $form = array_merge($form, get_edit_component_base_form_elements($form, 'xml_tag'));

  // form field to edit attributes, available attributes for command includes:
  $form['extra']['attributes']['''] = [];

  // grab populated data from 'extra' column from webform_component table and
  // fill it as default values for edit component form fields.
```

(continues on next page)

(continued from previous page)

```
  edit_component_form_fields_default_value($form);
}
```

Step 3.3: edit form elements for xml tag attributes.

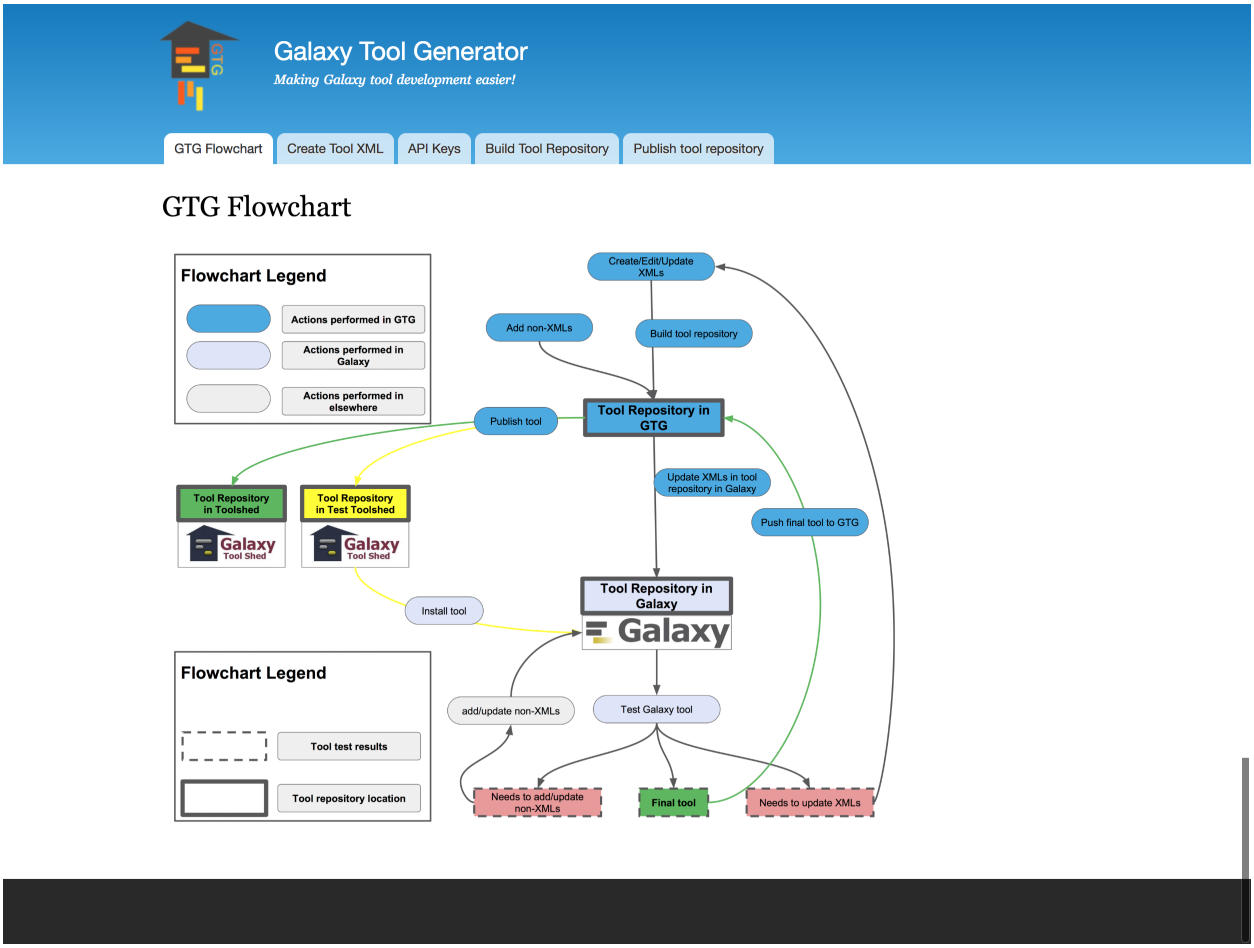
Below is the form definition function for creating the form of editing webform components. Edit this function to create form elements for each XML attributes.

```
/**
 * Implement edit command function.
 */
function edit_component_component_template(&$form) {
  unset($form['validation']);
  unset($form['display']);

  $form = array_merge($form, get_edit_component_base_form_elements($form, 'xml_tag'));

  // form field to edit attributes, available attributes for command includes:
  $form['extra']['attributes']['''] = [];

  // grab populated data from 'extra' column from webform_component table and
  // fill it as default values for edit component form fields.
  edit_component_form_fields_default_value($form);
}
```

CHAPTER 5

What is Galaxy Tool Generator (GTG)?

GTG is a Drupal based web application which enables developing and publishing Galaxy tools through web interfaces. Use the provided docker container to launch a site running tool generator. build your tool, and publish it to the Galaxy Tool Shed!


Galaxy Tool Generator
Making Galaxy tool development easier!
GTG Flowchart
Create Tool XML
API Keys
Build Tool Repository
Publish tool repository
GTG Flowchart
Flowchart Legend
Actions performed in GTG
Actions performed in Galaxy
Actions performed in elsewhere
Create/Edit/Update XMLs
Add non-XMLs
Build tool repository
Publish tool
Tool Repository in GTG
Update XMLs in tool repository in Galaxy
Push final tool to GTG
Tool Repository in Toolshed
Tool Repository in Test Toolshed
Galaxy Tool Shed
Install tool
Tool Repository in Galaxy
Galaxy
Flowchart Legend
Tool test results
Tool repository location
add/update non-XMLs
Test Galaxy tool
Needs to add/update non-XMLs
Final tool
Needs to update XMLs