

mc2 Gateway Engine Manual

Introduction

The purpose of the mc2 toolset is to provide the simplest environment as possible for configuring and accessing services, without any development or deployment effort by the scientific application developer. Moreover, the toolset offers some specific features not commonly found in the aforementioned solutions and which can be easily enabled or disabled individually for each customer through configuration. Among such features we mention: (i) file sharing between gateway users, (ii) support for provenance tracking of experiments' data inputs and outputs, and (iii) support for restricted anonymous access to gateway services.

Configuring the Tool

In order of automatically building a new scientific gateway the mc2 gateway engine has a set of XML configuration file that describes its functionalities and uses a properties file composed of *key=value* parameters that are used to configure the services that the gateway will access. Figure 1 illustrates the engine startup that follows the steps (i) reads the set of XML files and get information about the applications and modules that will be enabled in the gateway; (ii) receives the properties file and forwards it to the configuration service of the facade, that will configure all mc2 gateway required services; (iii) with the application information read from then XML files and the facade services configured, gets information about the application from the application service and builds the web interface automatically.

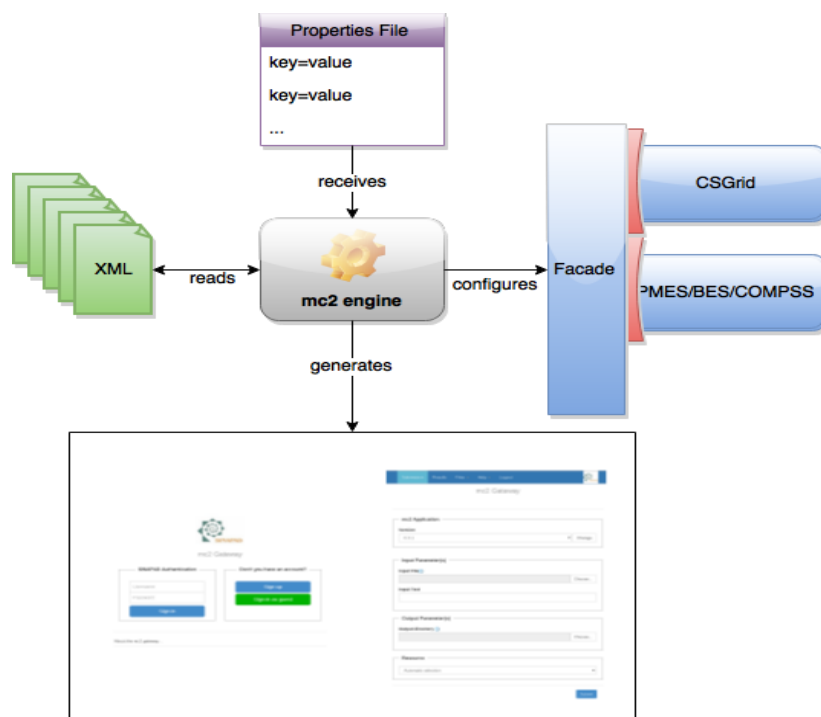


Figure 1 - mc2 Gateway Engine

The XML Files

The mc2 gateway tool is composed of four XML files:

1. **authentication.xml** – responsible for configuring the authentication methods of the gateway;
2. **modules.xml** – responsible for configuring the modules that will be enabled in the gateway;
3. **portal.xml** – responsible for configuring the applications that will be executed by the gateway;
4. **email.xml** – responsible for configuration the notification messages that will be sent by the gateway.

Each of the XML files has a XSD file that defines its structure on the way they can be understood by the gateway engine.

The authentication.xml File

The authentication.xml file is illustrated in Figure 2. There are three types of authentication supported by the mc2 gateway engine: LDAP, VOMS and Shibboleth.

The LDAP and VOMS authentication will be enabled just by setting the attribute **enabled** to **true**, because all information need for connecting with these two services will be configured into the facade concrete implementation. On the other hand, the Shibboleth authentication will request for two other elements: the **url** and the **storage-path**. The **url** element is the address where the Shibboleth authentication service is hosted and the **storage-path** is where the mc2 engine tool will look for the Shibboleth users tokens, that must be generated by the service hosted in the **url** provided on the configuration.

```
<authentication xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../xsd/authentication.xsd">
  <ldap enabled="true" />
  <voms enabled="true" />
  <shibboleth enabled="true">
    <url>shibboleth.authentication.url</url>
    <storage-path>shibboleth.token.storage.path</storage-path>
  </shibboleth>
</authentication>
```

Figure 2 - authentication.xml File

The modules.xml File

The modules.xml file is illustrated in Figure 3. There are four additional modules that may be enabled on mc2 gateway engine: Shared Files, Public Files, Guest User and Bulk Jobs.

The Shared Files module is responsible for allowing the gateway uses to share files between each other. The files that are shared will be available in a shared area that every authenticated user will have access and will be able to download them. Notice that guest users will not be able to access this shared area, since they are not authenticated into the gateway (as described below).

The Public Files module is responsible for allowing the gateway users to publish their results in the Internet and make them available for everyone. Different from the shared area, the public area cannot be accessed by the users directly. Only the users that have the public URL of the files will be able to download them. This URL is generated in the moment that a user chooses to publish a file and is also e-mailed for the user. After that he/she can choose what to do with the URL.

The Guest User module is responsible for allowing access to the gateway by anyone. Since this feature does not required any information about the user, some limitations such as file upload restrictions and the requirement of a captcha validation are automatically added to the user session. After enabling this module it is also possible to restrict the versions of the applications that will be available for the guest user. The "ALL" string is used if there are no version restriction and the applications versions must be separated by a semicolon character (;).

The Bulk Jobs module is responsible for allowing the submission of bag-of-tasks that vary only in fell parameters. The parameters of a bulk job will receive some additional parameters such as a begin value, an end value and a step value that may be informed by the user in the time of the submission or configured as a static value. These values may also have a minimum and maximum value that will be validated on the submission time. Since the gateway does not know how these values will be used by the application it is also possible to set the label of the parameters. This label may be set as a static text or as and Struts 2¹ global resource properties key. After submitting a bulk job another parameter will be automatically added to the application. This parameter is the current index of the loop that can be used as a controller to each task of the bag-of-tasks.

```
<modules xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../xsd/modules.xsd">
  <shared-files>
    <enabled>true</enabled>
  </shared-files>
  <public-files>
    <enabled>true</enabled>
  </public-files>
  <guest-user>
    <enabled>true</enabled>
    <allowed-versions>ALL</allowed-versions>
  </guest-user>
  <bulk-jobs>
    <enabled>true</enabled>
    <begin-value static="false" max="10">
      <label>struts.properties.key</label>
      <default-value>1</default-value>
    </begin-value>
    <end-value static="false" min="0" max="10">
      <label>End Value</label>
      <default-value>10</default-value>
    </end-value>
    <step-value static="false" min="1">
      <label>Step Value</label>
      <default-value>1</default-value>
    </step-value>
  </bulk-jobs>
</modules>
```

Figure 3 - modules.xml File

¹ <http://struts.apache.org/>

The portal.xml File

The portal.xml file is illustrated in Figure 4. This file describes the submission interface, including the applications that the gateway will execute and some minor functionality. The portal element contains four attributes: **allow-multiple-versions**, **auto-generate-interface**, **allow-resource-choice** and **result-check-interval**.

The **allow-multiple-versions** attribute defines if the gateway will allow users to select the versions of an application, otherwise the default version of the application will always be executed. The **auto-generate-interface** attribute defines if the gateway will automatically generate the submission interface every time it is redeployed. Notice that the first time the gateway is deployed its interface is always generated, but if **auto-generate-interface** is set to **false**, designers may make some layout changes and it will not be replaced on the next deployment. The **allow-resource-choice** attribute defines if the gateway users will be able to select in which resource the application will be executed, otherwise the resource will always be automatically chosen. The **result-check-interval** attribute defines the time interval, in milliseconds, that the gateway will be verifying if the submitted jobs have finished. After checking, if the job has finished, a notification will be sent to the user on his/her e-mail. Notice that this parameter may change depending on the average time that the application takes to execute.

The other elements of the portal.xml files are: the **acronym-name**, **full-name**, **project-name** and the **applications**.

The **acronym-name** and **full-name** elements are two strings that will appear on the gateway header. The **project-name** element defines the project that is bound to the gateway, and will define (along with the user credentials) if the user has access to the gateway, since he/she must have a project area to store his/her files. The **applications** element defines the applications that can be executed by the gateway and is composed by three other elements: **name**, **default-version** and **info**.

The **name** element defines the application name. This name will be used on the application service, to retrieve the application parameters for all its versions that will be used to generate the gateway submission interface. The **default-version** element defines the application default version that will be selected by default right after the user login. This element also defines the version of the application that will always be executed in gateways that the **allow-multiple-versions** attribute is set to **false**. The **info** element is a text free (or Struts 2 global resource key) information about the application that will be shown in the submission interface.

```
<portal xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../xsd/portal.xsd"
  allow-multiple-versions="true" auto-generate-interface="true"
  allow-resource-choice="true" result-check-interval="40000">
  <acronym-name>mc2</acronym-name>
  <full-name>mc2 Gateway</full-name>
  <project-name>mc2</project-name>
  <applications>
    <application>
      <name>mc2</name>
      <default-version>0.0.1</default-version>
      <info>mc2 Application</info>
    </application>
  </applications>
</portal>
```

Figure 4 - portal.xml File

The email.xml File

The email.xml file is illustrated in Figure 5. This file configures the SMTP service that will be used to send the job finished and error notifications. The element **smtp-host** contains the SMTP address that can be accessed by the gateway. The **email-from** and **email-to** defines the e-mail from which the users will receive the notifications and the e-mail to where the errors will be sent, respectively. The **job-finished-notification-message** element is composed by two other elements: **subject** and **body**. These elements are two strings (or Struts 2 global resource keys) that will compose the e-mail that will be sent to the user after one of their jobs has finished, together with the job id and the job status.

```
<email xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../xsd/email.xsd">
  <smtp-host>smtp.service.url</smtp-host>
  <email-from>sinapad-sup@lncc.br</email-from>
  <email-to>sinapad-sup@lncc.br</email-to>
  <job-finished-notification-message>
    <subject>Notification from mc2 gateway</subject>
    <body>You are receiving this message because one of your jobs has finished.</body>
  </job-finished-notification-message>
</email>
```

Figure 5 - email.xml File

The Properties File

After configuring the XML files, the scientific gateway developer will also have to set the correct properties to the mc2 gateway engine. These properties must be created in a *text/plain* file with the *key=value* format. The properties on this file will depend on the facade concrete implementation that the mc2 gateway will access. The properties file name that mc2 gateway engine expects is **portengin.properties** and must be in the directory structure illustrated next section.

The mc2 gateway contains a single property (illustrated in Table 1) that defines what facade implementations it must look for. This property is optional if the facade implementations are set in the **spring.xml** file on the correct mc2 gateway directory structure.

portengin.facade.impl	The ID of the face: bes or csgrid
-----------------------	---

Table 1 - mc2 Engine Properties

The CSGrid facade implementation requires some attributes for properly working, illustrated in Table 2.

openbus.host	The address where the Openbus is hosted
openbus.port	The port which the Openbus is listening
openbus.csgrid.offers.entity	The name of the Openbus offer entity
openbus.ldap.domain	The ID of the domain for LDAP on Openbus
openbus.voms.domain	The ID of the domain for VOMS on Openbus
csgrid.entity	The CSGrid entity of the gateway on Openbus
csgrid.cert	The Openbus certificate path of the CSGrid
ldap.host	The LDAP host
ldap.port	The LDAP port

ldap.dn	The LDAP DN
---------	-------------

Table 2 - CSGrid Properties

The PMES/BES/COMPSS facade implementation requires some attributes for properly working, illustrated in Table 3.

bes.address	PMES/BES web service address
bes.credential.user	PMES/BES username
bes.credential.password	PMES/BES password
bes.cdmi.address	CDMI service address
bes.cdmi.username	CDMI service username
bes.cdmi.password	CDMI service password
ldap.host	The LDAP host
ldap.port	The LDAP port
ldap.dn	The LDAP DN

Table 3 - PMES/BES/COMPSS Properties

The mc2 Gateway Structure

The Figure 6 illustrates the directory structure of a mc2 gateway. This structure must be followed by the gateway developer or mc2 gateway engine will not be able to find the required files.

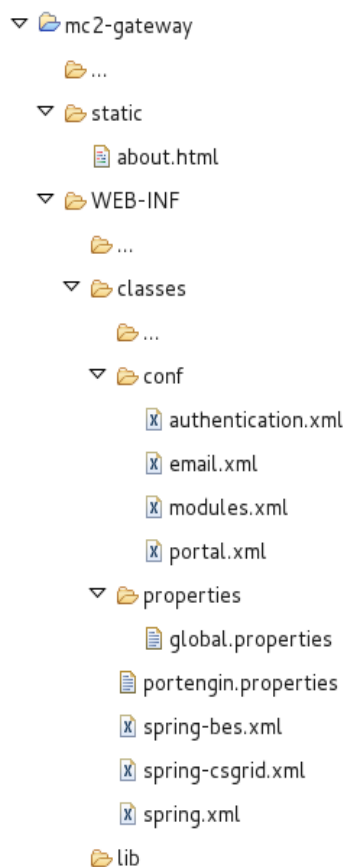
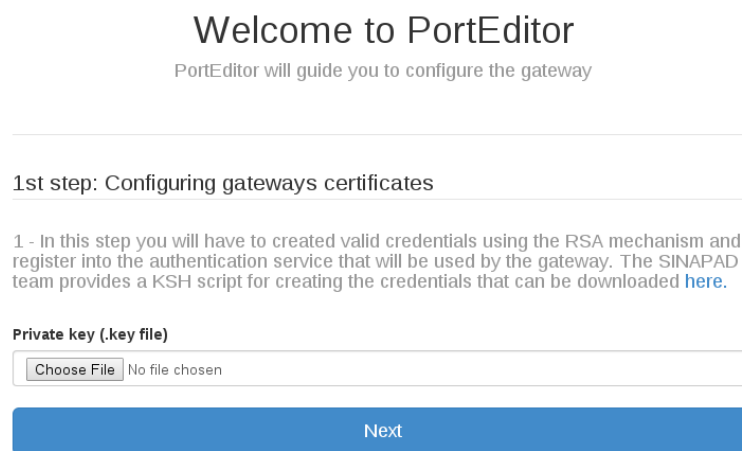


Figure 6 - mc2 Gateway Directory Structure

The mc2 Gateway Editor

The mc2 gateway engine works as a Web application and once it is deployed it automatically verifies if it is a new instance and provides a visual way for configuration itself. This visual interface is called mc2 gateway editor tool, and aims at aiding the scientific application developer in the edition of the XML files that set up the gateway engine.

The first screen of the mc2 gateway editor tool is illustrated in Figure 7. At the first step the editor tool will ask for a RSA private key that will be used by the gateway engine to authenticate the “PortEngin” user and access the facade services. Notice that the public key must be stored somewhere where the facade concrete implementation is able to find it, since it will be used to authenticate this user.



Welcome to PortEditor
PortEditor will guide you to configure the gateway

1st step: Configuring gateways certificates

1 - In this step you will have to created valid credentials using the RSA mechanism and register into the authentication service that will be used by the gateway. The SINAPAD team provides a KSH script for creating the credentials that can be downloaded [here](#).

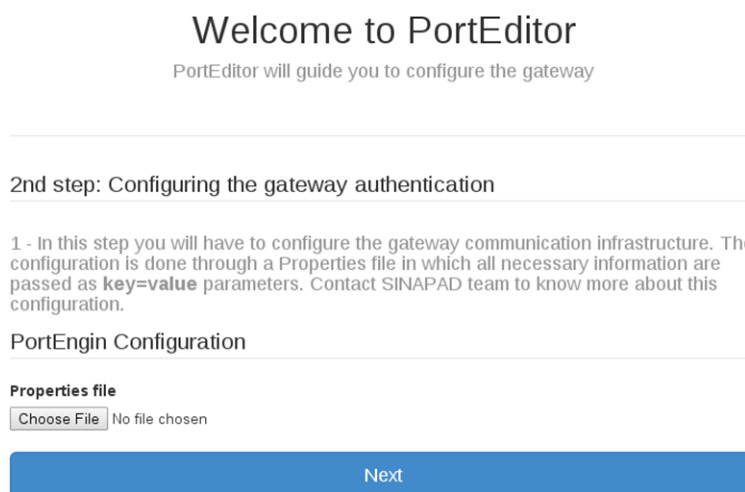
Private key (.key file)

No file chosen

Next

Figure 7 - mc2 Gateway Editor 1st Step

The second screen of mc2 editor tool is illustrated in Figure 8. At the second step the gateway developer must provide the properties file containing the information for connecting to the facade concrete implementation.



Welcome to PortEditor
PortEditor will guide you to configure the gateway

2nd step: Configuring the gateway authentication

1 - In this step you will have to configure the gateway communication infrastructure. The configuration is done through a Properties file in which all necessary information are passed as **key=value** parameters. Contact SINAPAD team to know more about this configuration.

PortEngin Configuration

Properties file

No file chosen

Next

Figure 8 - mc2 Gateway Editor 2nd Step

The third screen of the mc2 editor tool is illustrated in Figure 9. The third step will only be shown the private key and the properties file are correct, since the “PortEngin” user will connect to the façade to get information about the applications that are available.

Welcome to PortEditor

PortEditor will guide you to configure the gateway

3rd step: Configuring the gateway parameters

1 - In this step you will have to configure the gateways paramaters, including which executable it will run and which of the modules will be enabled.

Gateway Configuration

Acronym Name

mc2

Full Name

mc2 Gateway

About (HTML accepted)

Here comes a text about the gateway

☐ Allow multiple versions

☒ Allow resource choice

Application Configuration

mc2

1.0.0

Modules Configuration

☒ Shared files

☐ Public files

☐ LDAP login

☒ VOMS login

☐ Restricted anonymous access

E-mail Configuration

Subject

Notification from mc2 gateway

Body

You are receiving this message because one of your jobs has finished.

Next

Figure 9 - mc2 Gateway Engine 3rd Step

After filling in all the requested information the developer will proceed to the last step, illustrated in Figure 10, where the gateway will be successfully configured and a link to access it will be available.

Welcome to PortEditor

PortEditor will guide you to configure the gateway

Gateway configured!

You have just configured the gateway. Click [here](#) and enjoy ;-).

Figure 10 - mc2 Gateway Editor Last Step

Notice that not all the options available in the XML files are covered by the mc2 gateway editor, since it is supposed to make it easier to configure a simple gateway. For gateway that have more complex behavior and/or may want to have some layout changes, a manual configuration step on the XML and HTML files is also needed.