RMQ Publication

The RabbitMQ module bundle provides an alternative way of publishing content and synchronizing instances in Magnolia. RabbitMQ is open source message broker software (a.k.a message-oriented middleware) that implements the Advanced Message Queuing Protocol (AMQP).

The Magnolia RabbitMQ modules are suitable for:

- Environments consisting of more than three public instances.
- Environments that need to synchronize with other environments.

The RabbitMQ alternative gives you full control over publication and synchronization, while reducing the load on your author environment. Publication is persisted into queues, allowing for easier setup of continuous deployment environments. It's also possible to audit publication state using the app provided by the monitoring module.

JIRA	QAARQ
Git	rabbitmq-publication

Installation

Maven is the easiest way to install the modules. Add the following dependencies to your bundle:

```
<!-- Integrates RabbitMQ with Magnolia -->
<dependency>
  <groupId>info.magnolia.rabbitmq</groupId>
  <artifactId>magnolia-rabbitmq-connector</artifactId>
  <version>${rabbitmqVersion}</version>
</dependency>

<!-- Allows you to synchronize and activate over RabbitMQ -->
<!-- Contains classes for both activation and publishing modules -->
<dependency>
  <groupId>info.magnolia.rabbitmq</groupId>
  <artifactId>magnolia-rabbitmq-activation</artifactId>
  <version>${rabbitmqVersion}</version>
</dependency>

<!-- Allows for monitoring of activation from author -->
<!-- Not a required module for publishing -->
<dependency>
  <groupId>info.magnolia.rabbitmq</groupId>
  <artifactId>magnolia-rabbitmq-monitoring</artifactId>
  <version>${rabbitmqVersion}</version>
</dependency>
```

Installing RabbitMQ

Follow the instructions at RabbitMQ to download and install the service. Instructions and packages are provided for all operating systems. It's also a good idea to enable the management plugin.

To get started even quicker consider using CloudAMQP to get RabbitMQ as a service. For more detailed instructions see RMQ Setup Example.

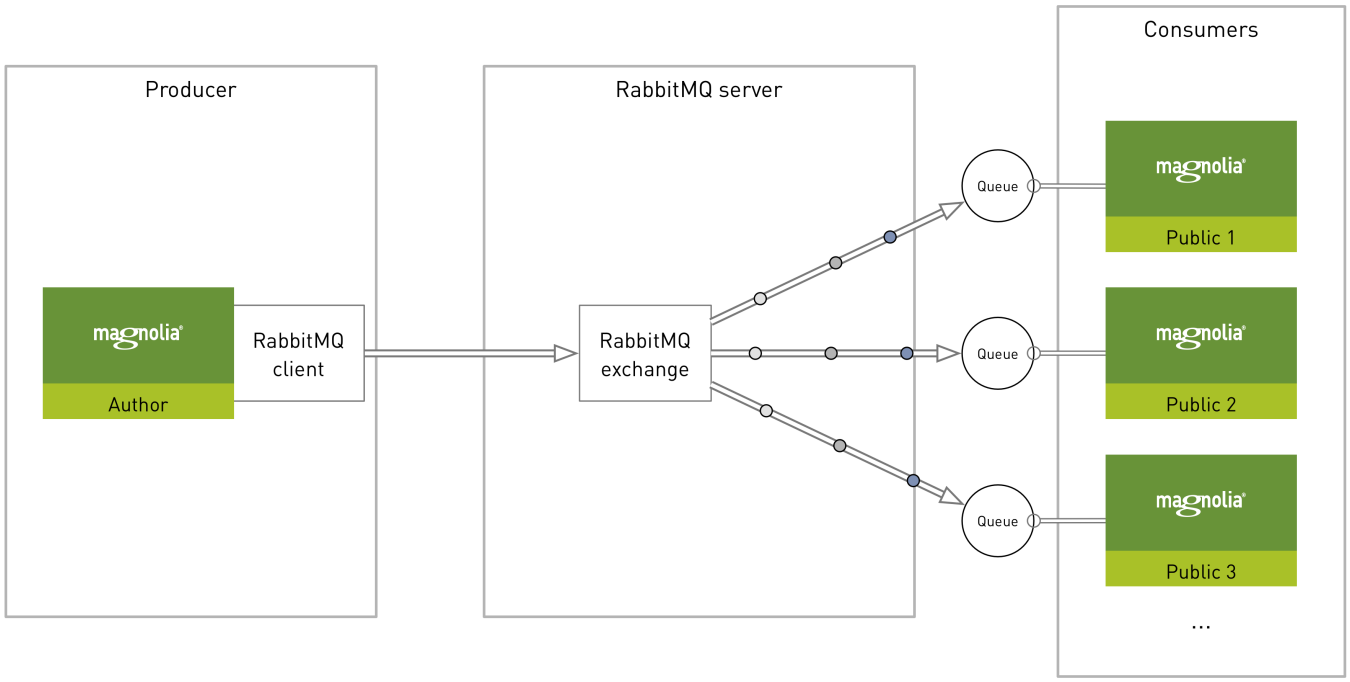
Versions

1.9	**Magnolia 6.1**
1.8	**Magnolia 5.7**
1.7.1	**Magnolia 5.6**

Usage

RabbitMQ can be used for publication and/or synchronization. There exists a single exchange sever with a fanout queue. Exchanges are configured in the connector module.

The publication setup would look like this:



Configuration

The exchange is configured on both author (producing instance) and public (consumer instance).

Client

The client configuration is used to integrate Magnolia with an instance of RabbitMQ.

Example: `/modules/rabbitmq-connector/rabbitmq-client/sampleClient`

`clientName`	**required** The value specified here must match the `clientName` property in the activate and deactivate commands for RabbitMQ. Example: `/modules/activation/commands/withContentSyncingVersioned/activate/rabbitmq-activate@clientName`

`enabled`	**required** Enables and disables the client. Toggling this property restarts the client.
`hostName`	**required** Address of the broker.
`password`	**required** Password of the user account to connect with.
`portNumber`	**required** Port number to connect the AMQP service.
`username`	**required** Username of the user account to connect with.
`virtualHost`	**required** Virtual host to connect to.

Exchange

The exchange is the kind of router that the queues connect to. The producer instance pushes messages to the exchange and the exchange then decides what to do with them. You must configure the exchange configuration on the author or producing instance.

Example: `/modules/rabbitmq-connector/rabbitmq-client/sampleClient/exchangeConfig`

`name`	**required** The name specified here must match the `exchangeName` property in the activate and deactivate commands for RabbitMQ. Example: `/modules/activation/commands/withContentSyncingVersioned/activate/rabbitmq-activate@exchangeName`
`type`	**required** Exchange type. See AMQP 0-9-1 Model Explained for more. Options: `fanout`, `direct`, `topic`, `headers`

Queue

A list of queue definitions to bind with the exchange should be configured under the node `queueConfigList`.

Example: `/modules/rabbitmq-connector/rabbitmq-client/sampleClient/exchangeConfig/queueConfigList/queue`

`autodelete`	**optional** Automatically deletes the queue when last consumer unsubscribes.
`exclusive`	**optional**, *default is `false`* Used for only one connection. Queue is deleted when that connection closes.
`name`	**required** Name of the queue.
`routingKey`	**optional** Key used to route messages. The queue binds to the exchange with the routing key. Applicable when `direct` mode is enabled in the exchange.

Consumer

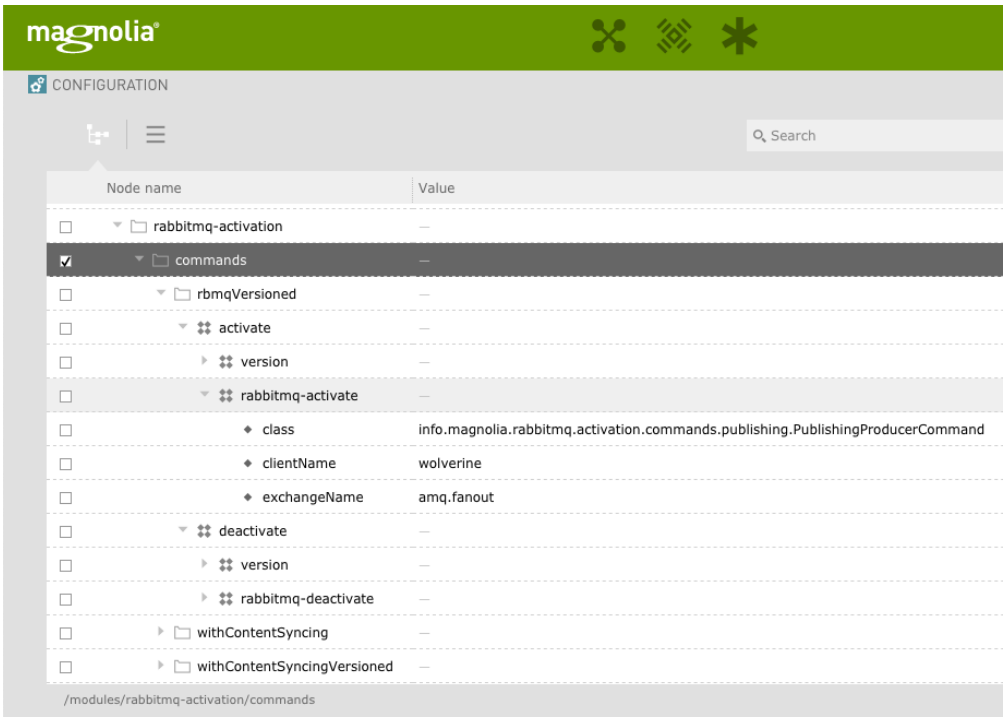
Consuming messages on a public instance is called activation or publication.

Example: `/modules/rabbitmq-connector/rabbitmq-client/sampleClient/consumerDefinition/activationConsumer`

`ackExchangeName`	**optional** Used for monitoring. An ACK exchange is bound to a queue on which activation confirmation messages are sent.
`autoAck`	**optional**, *default is false* Automatic acknowledgement.
`clientName`	**required** Name of the client to use.
`consumerClass`	**required** Consumer class which implements AbstractMQConsumerJob . Example: `info.magnolia.rabbitmq.activation.jobs.ActivationConsumer`
`enabled`	**required** Enables and disables the consumer.
`name`	**required** Name of the consumer.
`notifyUserOnStatus`	**optional**, *default is false* Notify the user of the status.
`testMode`	**optional**, *default is false* Set the consumer for test mode.
`updatePolicy`	**optional**, *default is UPDATE_STRICT* Options are UPDATE_STRICT and UPDATE_MERGE.
`verifyAuthorConsumer`	**optional**, *default is true* Authorizes the signature and data of message from other instance.

Commands

The RabbitMQ Activation module installs the activation commands here: `/modules/rabbitmq-activation/commands`.



Catalogs

You can select commands/catalogs to match your purpose for using RabbitMQ.

Here are a list of the catalogs with a description of their purpose. Inside each catalog folder you will find a "command chain" node which chains together the commands that should be performed.

Catalog	Command Chain	Description
`withContentSyncing`	`activate` `deactivate`	Allows you to hook into transmission over RabbitMQ once you have done a standard activation. Calls the standard activation command first and then calls `rabbitmq-activate` or `rabbitmq-deactivate` command.
`withContentSyncingVersioned`	`activate` `deactivate`	Same as `withContentSyncing` but adds the versioning command to the chain. Versioning will be performed first in the chain.
`rbmqVersioned`	`activate` `deactivate`	Replaces standard activation with RabbitMQ versioned activation. Calls the versioning command first and then calls `rabbitmq-activate` or `rabbitmq-deactivate` command.

The commands themselves `rabbitmq-activate` and `rabbitmq-deactivate` have the following configuration options.

`clientName`	**required** Configured client name. The `clientName` property must be the name of a configured RabbitMQ client**.**
`exchangeName`	**required** Configured exchange name. The `exchangeName` property must be the exchange name of a configured Rabbit MQ client.

`delegateAck`	**optional** , *default is* `true` When set to `true` message acknowledgements are delegated to RabbitMQ. Set to `false` if you want to use the rabbitmq-monitoring module instead.
`stackSize`	**optional** , *default is* `1` Stack size allows you to send more nodes per activation message, making messages bigger but reducing the number of messages in the queue.
`blocktime`	**optional** , *default is* `100` Blocktime in seconds. Works only if `delegateAck` is `false` and if the ack exchange is configured correctly.
`blockActivation`	**optional** Blockactivation. Works only if `delegateAck` is `false` and if the ack exchange is configured correctly.
`minNbrOfPublics`	**optional** Minimum number of public instances to wait ack for. Works only if `delegateAck` is `false` and if the ack exchange i s configured correctly.

Activation mechanism

Each activation is prefixed with a BOA (Beginning Of Activation) message and post-fixed with a EOA (End Of Activation) message. When the consumer receives a message on a public instance, the message is stamped with the identity of the receiver and send back to the ACK queue. When the author receives the message, the author knows exactly if the message started consuming the activation. In addition, identifying the activation in queue with BOA and EOA allow identification of TX inside the the queue.

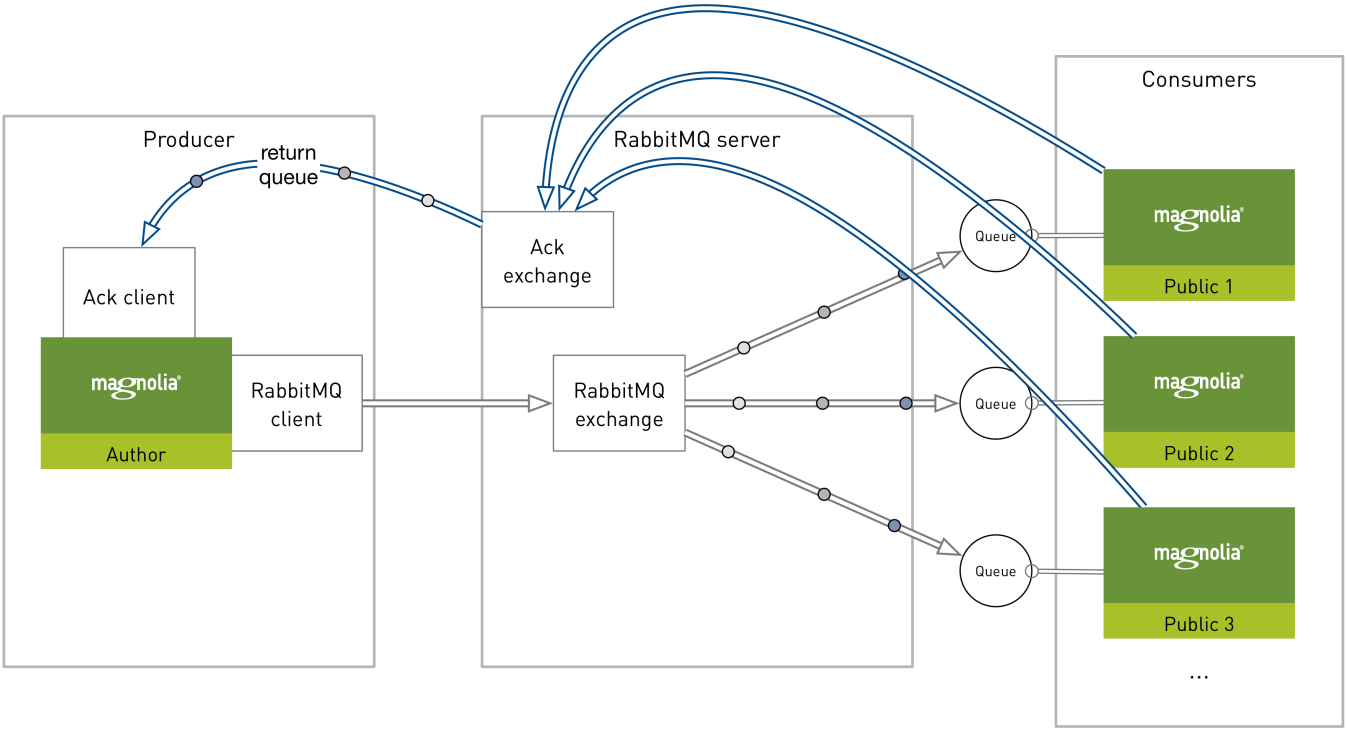
Monitoring

RabbitMQ monitoring allows for public instances to confirm publication with the author instance. In this setup author and public both play the producer /consumer role. This is an optional configuration, known as acknowledgment, for those wishing to audit publication status.

For an acknowledgement setup:

- A simple fan out pattern is used for activation from one Magnolia author instance to N public instances.
- Acknowledgements are **not** delegated to RabbitMQ. Instead, notifications of activation success or failure are sent back to the acknowledgement (ACK) exchange for consumption by author.
- The AMQP 0-9-1 (Advanced Message Queuing Protocol) messaging protocol is used.

The publication w/ acknowledgement setup would look like this:



Public monitoring app

The Public Monitoring app displays the states of any instance that is consuming on a queue. The app will show information messages containing the status of activation grouped by the user who created the request.

Activation states:

- **IOA** – Message in queue not yet received by consumer but sent by author.
- **BOA** – Beginning of activation, message received by consumer, nodes will be consumed shortly after.
- **EOA** – End of activation, message consumed and remitted after all nodes were successfully consumed.
- **EXA** – Exceptions on activation, nodes were not processed correctly and the consumer sent the exceptions to the author before shutting itself down.

> If an activation fails, the message is put back into the queue and an exception is pushed into the acknowledgment queue. The consumer is then stopped allowing you to fix the problem and to remove the faulty instance from the load balancer.

REST endpoints

REST endpoints are used for controlling and monitoring publication.

> By default the endpoints are not enabled. They must be explicitly configured through the Security app to be available to the outside world.

The monitoring module provides three endpoints.

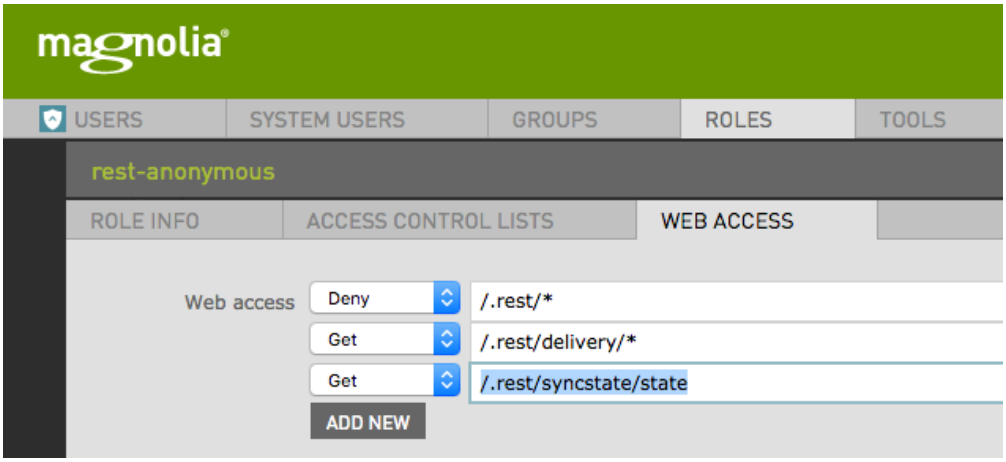
- Synchronization State
- Public monitoring
- Client control

Synchronization State

The SyncState REST service gets the current state of publication immediately (without checking the ACK return queue).

> This service stores values in properties on the node `/modules/rabbitmq-activation/syncNode`. These properties should **not** be changed by hand. They are **not** configuration settings.

To enable the SyncState service for anonymous access add the following permissions to the `rest-anonymous` role:



You can call the REST endpoint using cURL:

```
curl http://<hostname>:<port>/<context>/.rest/syncstate/state
```

Example output:

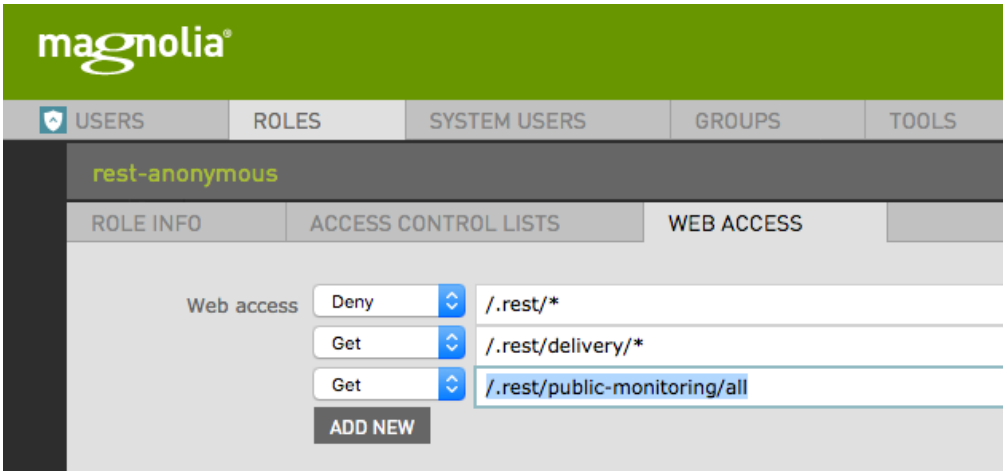
```
{"seqNbr":2011,"stamp":1467280192928,"topoTag":""}
```

Public monitoring

The public monitoring REST service gets the results of all consuming instances returning messages on the ACK queue. This service is used by the Public Monitoring app.

> You can access the monitoring screen at http://<hostname>:<port>/<context>/.resources/magnolia-rabbitmq-monitoring/webresources/visualisations/public_monitoring.html or in the Public Monitoring app.

To enable the public monitoring service for anonymous access add the following permissions to the `rest-anonymous` role:



You can call the REST endpoint using cURL:

```
curl http://<hostname>:<port>/<context>/.rest/public-monitoring/all
```

Example output:

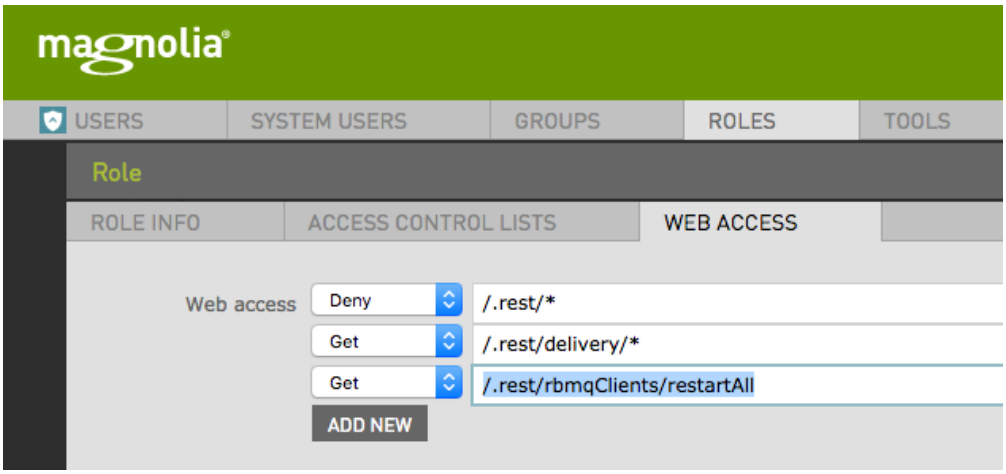
```
[{"workspace":"dam","address":"192.168.10.85","name":"administrators-MacBook-Pro.local","syncStamp":"
1466422838236","diff":"0","uuid":"2acd7055-7168-4926-b66a-47400ab78d2e","seqNbr":"1977","exceptions":"{}"}]
```

`workspace`	Workspace of last activated node.
`address`	IP of consuming instance.
`name`	`hostName` of consuming instance.
`syncStamp`	Timestamp of last activated node.
`diff`	Difference on `seqNbr` with other instances who activated a similar node.
`uuid`	UUID of last activated node.
`seqNbr`	Sequence number of last activated node.
`exceptions`	Exceptions that may have occurred.

Client control

The client control REST service allows you to restart the consumer on the remote instance.

To enable the client control service for anonymous access add the following permissions to the `rest-anonymous` role:



You can call the REST endpoint using cURL:

```
curl http://<hostname>:<port>/<context>/.rest/rbmqClients/restartAll
```

Warnings

- This module is at INCUBATOR level.
- Acknowledgements are delegated to RabbitMQ. At this time notification of activation success or failure is not sent back to the author instance.
  See QAARQ-18 - Getting issue details... STATUS .
- Users of previous versions:
  - This module has been updated to use the new Publishing modules. Update the configuration to use the new publication based classes. Have a look at the bootstrap files here for an example.
  - Be sure to remove the activation module from your configuration.
  - Be aware that the commands have moved from `/modules/activation/commands` to `/modules/rabbitmq-activation/commands`.
- RabbitMQ Publication is not design to accommodate workflow. You might want to consider removing it entirely from your build using an exclusion.
  See QAARQ-30 - Getting issue details... STATUS .

workflow exclusion

```
<dependency>
  <groupId>info.magnolia.eebundle</groupId>
  <artifactId>magnolia-enterprise-pro-webapp</artifactId>
  <version>${magnoliaBundleVersion}</version>
  <type>pom</type>
  <exclusions>
    <exclusion>
      <groupId>info.magnolia.workflow</groupId>
       <artifactId>magnolia-module-workflow</artifactId>
    </exclusion>
    <exclusion>
      <groupId>info.magnolia.workflow</groupId>
      <artifactId>magnolia-module-workflow-jbpm</artifactId>
    </exclusion>
    <exclusion>
      <groupId>info.magnolia.workflow</groupId>
      <artifactId>drools-jcr-persistence</artifactId>
     </exclusion>
     <exclusion>
       <groupId>info.magnolia.workflow</groupId>
       <artifactId>jbpm-jcr-persistence</artifactId>
     </exclusion>
  </exclusions>
</dependency>
```

Changelog

- Version 1.9
  - QAARQ-34 - Getting issue details... STATUS
- Version 1.8
  - QAARQ-31 - Getting issue details... STATUS
  - QAARQ-29 - Getting issue details... STATUS
  - QAARQ-28 - Getting issue details... STATUS
  - QAARQ-27 - Getting issue details... STATUS
  - QAARQ-26 - Getting issue details... STATUS
  - QAARQ-25 - Getting issue details... STATUS
  - QAARQ-24 - Getting issue details... STATUS
- Version 1.7.1
  - QAARQ-28 - Getting issue details... STATUS
  - QAARQ-26 - Getting issue details... STATUS
  - QAARQ-24 - Getting issue details... STATUS
- Version 1.7 - Initial release of the extensions version of the module.
  - QAARQ-22 - Getting issue details... STATUS
  - QAARQ-21 - Getting issue details... STATUS