# Overview

This document describes the operation of ActiveNet’s “multi-tenant servlet” system. This allows a single servlet to transparently serve requests from multiple orgs.

This functionality was built into Activenet in 2012, but was never quite finished, and was never deployed in multi-tenant mode. As a result, it is necessary to redocument its operation, and make some changes to the request processing to make it fit better with the new Jetty implementation.

Moreover, although the servlet was made fully multi-tenant aware at that time, because servlets have never received requests from more than one org, bugs have be introduced, in which the needed coding standards for multi-tenant operation were not followed.

# Reference integration model



## 1. Requests use existing URL structure

The original multi-tenant servlet implementation required different URL structures for multi-tenant servlets. I believe this was a result of limitations to the ServletExec mapping. This would require changes to web service clients, such as CUI, as well as breaking external links and bookmarks. This has now been changed to use the same URL structure as existing single-tenant servlets:

* /<orgname>/servlet/\*.sdi for standard servlet methods; e.g., /ymcala/servlet/adminlogin.sdi
* /<orgname>/<wsname> for web service requests’ e.g., /ymcala/ActiveNetWS

## 2. Only one servlet is deployed on each server

For the single-tenant Jetty installation, there is a servlet per orgsite, with a context path of the orgsite name. Each servlet is listening on a unique port.

For multi-tenant, there is a single servlet, listening on a single port. The context path must be empty (“/”).

[Comments from meeting]: Theoretically, multi-tenant servlet supports all orgs, but in early phase, the multi-tenant servlet will serve only orgs from same pool, from the deployment perspective, multi-tenant servlet will be deployed to side A and B for given pool, and serve orgs originally assigned to the pool.

## 3. The F5 forwards all requests on a server to a single port.

The F5 receives a requests and dispatches it to a server in the appropriate web server pool (data group). This operation is driven by an org mapping file, generated during deployment from the org configuration data in the ActiveNetSites DB. This file will require a port number for multi-tenant:

* For old-style, ServletExec implementation, the file only needed to map an org to an F5 datagroup.
* For Jetty, the file needs to map an org to a data group and a port
* For multi-tenant Jetty, all orgs in the pool have the same port number, so all requests go to the single servlet.

## 4. Jetty processes the request based on web.xml

Jetty receives the request, and based on entries in the Web.xml, first sends the request to one or more Filters, then to a servlet.

## 5. OrgContextFilter creates OrgContext for request

Based on the web.xml, all requests for org-specific URLs go to the OrgContextFilter. This identifies the org from the URL (e.g., "ymcala"), and finds or creates an OrgContext for that org.

The OrgContext must also be setup separately for any threads that are created or any threads used from a thread pool. Additionally, If JMS is used, the OrgContext needs to be setup for the onMessage methods.

Similarly, if there are other threads created by any other technology, they also need the OrgContext setup correctly.

## 6. All org-specific data is referenced via the OrgContext

The OrgContext is available to the rest of the servlet code via a ThreadLocal. All org-specific "static" data is referenced via the OrgContext. For example, the cached activities HashTable, rather than being a static of the Activity class, is referenced via the getActivityList() method of the OrgContext.

[Comments from meeting]: data cache by static fields need to move into OrgContext, a new annotation will be created to notate further static fields cache to explain reason for not in OrgContext, this will also be covered as a check point in code review and unit testing. Overall cache strategy will not be covered in early stage of multi-tenant conversion.

## 7. .ini file is parameterized

During initialization of an OrgContext, the servlet reads its configuration file (.ini) to get key information such as the JDBC URL. The .ini file, this information is parameterized, and when the servlet reads it, it makes substitutions into it based on the OrgContext. For example, in the JDBC URL, it substitutes %org\_site\_name% with the actual orgsite name (which is the database name).

## 8. Threads are started for each OrgContext

During OrgContext init, an instance of each background thread is created for the org, which remembers its OrgContext.

[Comments from meeting]: current background threads which are not OrgContext-aware need to be fixed to inherit from BackgroundThread or to make as OrgContext-aware.

# Areas of environment modification

In order to implement the above model, the following changes need to be made to the deployment / environment:

* **Servlets:** If a pool is flagged multi-tenant, create a single Jetty instance on that server.
* **Org mapping file:** In the org mapping file generation, route requests for all orgs in a MT pool to the same port (presumably a standardized port used by all multi-tenant servlets.
* **.ini file:** When the .ini file is generated by deployment, some entries which should be written with parameters instead of exact values, such as using %org\_site\_name% instead of the org name for the database name and URLs. Note that this new format should work for single-tenant implementations as well.
* **Web.xml:** The web.xml was modified, but again, it appears the new web.xml will work for ST and MT.

# Required coding standards

* All org-level data, such as cached data and global configuration, must be accessed via the OrgContext class, and not implemented as statics.
* (TBD) Standard for thread context.
* If Possible you should create checkstyle rules that identify improper access.

# Areas still requiring development testing:

* WS calls are properly handled based on SoapUI testing, but Flex resource scheduler is generating an invalid incoming WS URL (/ActiveNetWS instead of /org/ActiveNetWS). The assumption is that this is Flex code formatting the URL in a way which does work with single-tenant, but doesn’t provide the CUI and Flex UI need to be tested.
* All statics need to be reviewed to see if there are any which need to be moved to the OrgContext. In the course of normal development, a bug was recently found traced to a new static, so this is a real problem.
* It appears some of the threads don't currently work without a "default context", which is currently set as the first context created. The default context should be reviewed; I believe OrgContext should not have one, because this prevents correct multi-tenant operation.
* SSO servlet: I see in the web.xml that \*.sso requests are sent to a different servlet; this must be tested for multi-tenant operation.