

2805 Pontiac Lake Rd
Suite 2B
Waterford, MI 48328
Phone: (248) 706-1540
Fax: (248) 706-1002
Web: korsengineering.com

Poster Module for Niagara AX Use Guide

Version 1.0
2009-08-31

INSTALLATION

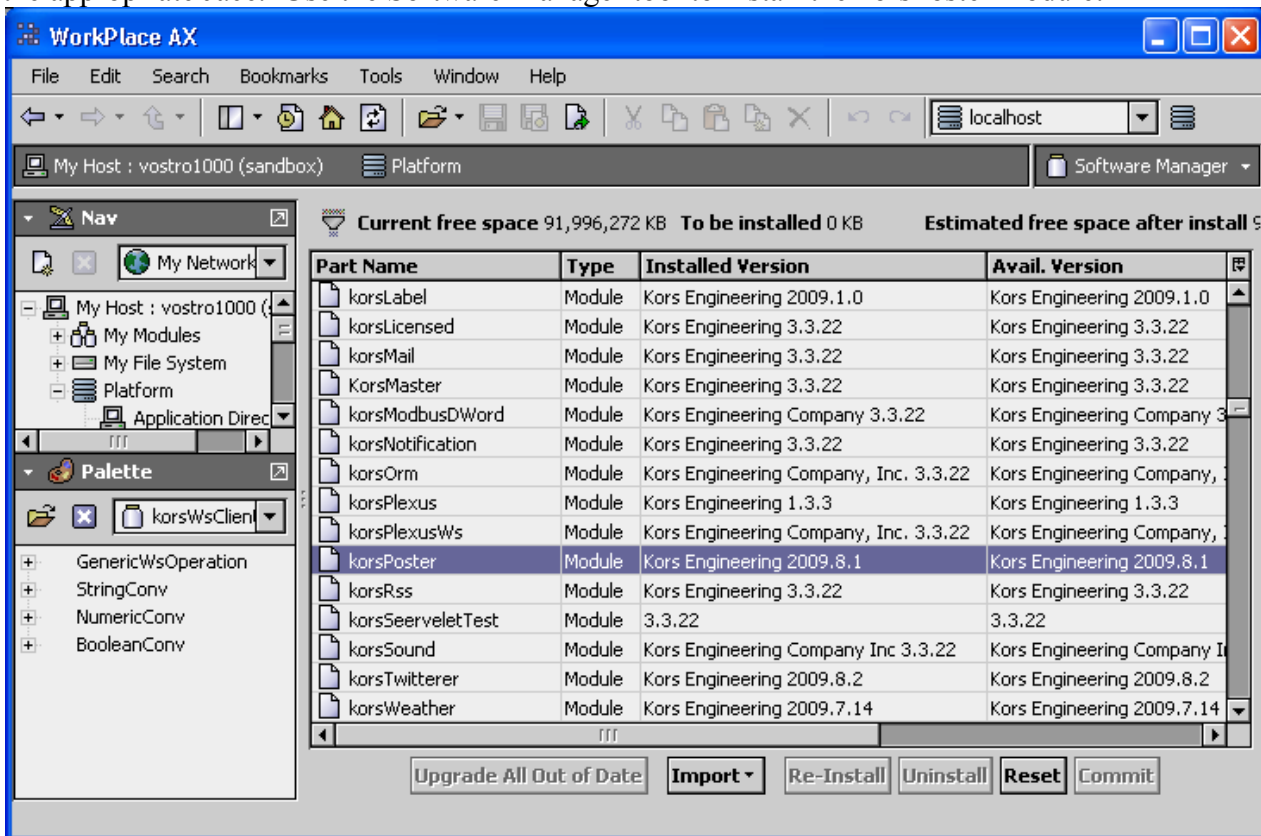
Installing the korsPoster module is done using Niagara WorkPlace AX. There are two files required for installation, the “jar” file that contains the korsPoster module, and the license file provided by Kors Engineering when you purchased the software. (When you don't have a valid license, the module runs in trial mode, and its components only send the first ten messages. In all other respects they operate the same way as during normal operation, so you can test them and see whether they fit your needs.)

OBTAIN THE MODULE

The first step in the installation is to obtain the [korsPoster.jar](#) file from the Kors Engineering website. Save this file on the computer running the WorkPlace AX software in the “modules” folder in your Niagara installation directory. For example, the default installation directory for AX 3.4.43 would be C:\niagara\niagara-3.4.43\modules. Once you have saved the file, you will have to restart WorkPlace AX for it to find the new file.

COPY THE MODULE TO YOUR JACE

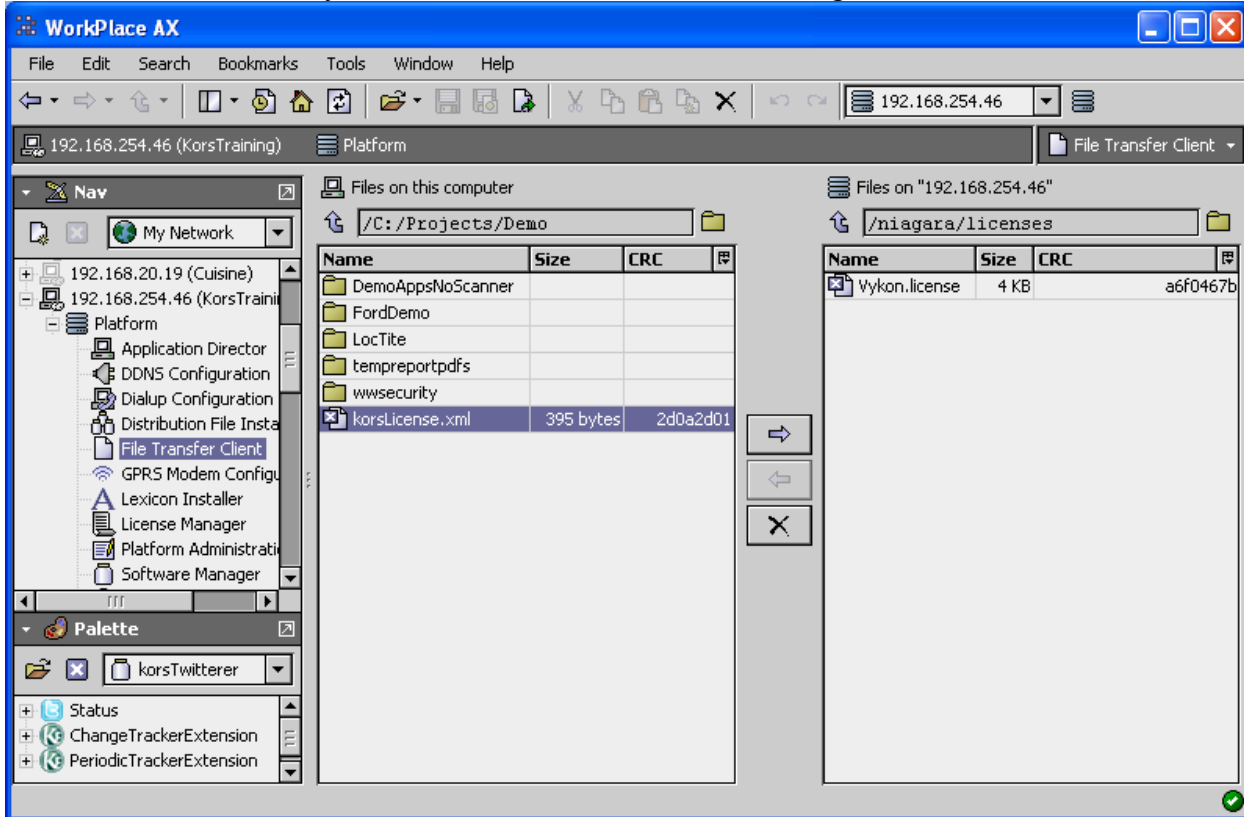
To deploy this file to running Niagara AX Jace units, use WorkPlace AX to connect to the Platform of the appropriate Jace. Use the Software Manager tool to install the korsPoster module:



You will have to restart the Jace once the module is installed.

LICENSE YOUR JACE

If you purchased a license for the module, you should have been emailed a korsLicense.xml file. Each Jace requires a unique license file, so it is important have the correct license file for each Jace. The license file is based on the Host ID of your Jace unit. The file is a simple text file, so if you need to find out which Jace a file is for, open it with notepad and the Host ID will be listed inside the file. To license a particular Jace, connect to the Platform of the Jace using the WorkPlace AX software. Use the File Transfer Client to copy the correct korsLicense.xml file to the /niagara/licenses folder in your Jace. You should restart your Jace for the new license to be recognized.



USING THE COMPONENTS

The module has three main components: ChangeTrackerExtension, PeriodicTrackerExtension and Sender. The first two are both designed to export information taken from a specific control point. However, they differ in the way they keep track of the point information. ChangeTrackerExtension attempts to export data whenever the output value of the point changes, while PeriodicTrackerExtension sends snapshots of the same data at user-defined time intervals. Finally, Sender is a much more generic component, which allows you to make HTTP requests with arbitrary content and headers.

CHANGE TRACKER EXTENSION

The Change Tracker Extension is available in the korsPoster Palette. This component is meant to be

added to an existing Niagara data point to send out an HTTP POST request whenever the output value of the “parent” point changes. Functionally, the object works like a History or Alarm extension. To use it, you need to drag-and-drop it to the target point, which can be done in Properties view, or via icons in the Nav panel. By default, this extension is configured to send data to Twitter servers – all you need to supply is user name and password. However, it can be used with a variety of different services.

Whenever this component sends out a request, it uses the Format field to generate the POST content. You can use Niagara's standard format syntax to make a template that will be automatically filled by the system each time. The plain text will be sent to the recipient without alteration, while the results of variable substitutions will be escaped using the specified Escape Method. (Currently, the extension supports URL encoding and XML entities.) You could, for example, use the component with a simple SOAP web service. The escape method would be set to XML and the Format could look like this:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:tar="targetNs">
  <soap:Header/>
  <soap:Body>
    <tar:SetStringPointRequest>
      <tar:pointId>%parent.displayName%</tar:pointId>
      <tar:value>%newValue.value%</tar:value>
    </tar:SetStringPointRequest>
  </soap:Body>
</soap:Envelope>
```

Component Fields	
Field Name	Description
URL	Full URL to send data to
User Name	HTTP authentication login (optional)
Password	HTTP authentication password (optional)
Format	Template for the body of the POST request. If you are not familiar with the Niagara formatting syntax, search the Niagara Central forums for examples and documentation. For this object, use %newValue% or %newValue.value% to embed the latest value of parent point into the message.
Escape Method	Substituted valued in the format field are escaped using one of the specified methods.
Enabled	Must be set to true to send requests
Active Period	Can be used to limit the times and days of the week to send requests
Active Status	Can be use to limit the point statuses that will send requests
Last Update Time	Time and date of the last successful request
Last Update Text	Content of the last successful request

Last Update Error	If an error occurred on the last post attempt, the error message will appear here
-------------------	---

THE PERIODIC TRACKER EXTENSION

Similar to the Change Tracker Extension, the Periodic Tracker Extension must be added to an existing Niagara point object to function. It is used to send requests with point data at specific time intervals.

All setup fields for the Periodic Tracker are the same as for the Change Tracker, with one new field, the “Period”. Use this field to set the interval at which the object will send new requests.

SENDER

Sender object is meant for sending various HTTP requests, having content composed somewhere else in the system. Its fields are mostly plain Java strings, so you would need to use a converter of some kind in case you want to send content of a status point. [Niagara AX Community Module](#) project has a free converter that can be used for such purposes.

Generally, you would fill all the appropriate slots and trigger “send” action to output the data. However, to simplify the wiring logic, you can use “Send on Change” option, that will automatically send out a request whenever Content or URL fields are updated.

The component also supports custom headers. However, that slot is hidden (since it takes space and is not likely to be used often). You can make it visible by going into slot sheet view, right-clicking on it, selecting “Config Flags”, and then unchecking “Hidden” in the window that opens.

Component Fields	
Field Name	Description
Method	HTTP method that will be used to submit the data. (E.g. GET, POST, HEAD, PUT, or DELETE.)
URL	The address to which the requests will be sent.
User Name	HTTP login.
Password	HTTP password.
Send on Change	If set to true, the component will send a request every time URL or Content fields are updated.
Content	Body of the request. The text is not escaped in any way.
Headers	(Hidden.) HTTP headers for the request formatted in a standard HTTP fashion. Headers are newline-separated. Header name is separated from the content by colon.
Last Send Time	The last time a request was successfully sent.

Last Response Text	Response content to the last request.
Last Send Error	If an error occurs during a request, its description will be here.