How to Use the dHTMLx Editable Grid within MII

Applies to:

SAP MII v12.1

For more information, visit the Manufacturing homepage.

Summary

The following document will discuss in detail how to create a working interface from the dHTMLx Editable Grid (http://www.dhtmlx.com/docs/products/dhtmlxGrid/) to MII Query Templates. It also includes a sample project with a working example and content to illustrate how this interface can be achieved. The project content can be downloaded from here.

Author: Salvatore Castro

Web Site: http://www.scastro.net/sam

Company: SAP Labs, LLC

Created on: 10 January 2010

Author Bio

Salvatore Castro of SAP Labs and has a Bachelors Degree in Computer Engineering and a Masters Degree in Computer Science both through the Rochester Institute of Technology. He is a member of the MII Product Management group under John Schaefer.

© 2010 SAP AG

Table of Contents

© 2010 SAP AG

Overview

The following document goes over how to interface a third party web application with the SAP MII software to create a very flexible and user-friendly Editable Grid. There are many subtle features that exist within the SAP MII product that make this type of solution very easy to achieve with only a small amount of effort. The project content has many examples of what the dHTMLx grid looks like and can do from their web site but the ThirdPartyTools/WEB/dhtmlxSuite/Grid/samples/mii_integration location contains relevant content for the SAP MII product along with an example of how to achieve this integration. Since everything in the SAP MII application can be accessed via a standard HTTP URL call commonly referring to as a REST interface, http://en.wikipedia.org/wiki/Representational_State_Transfer, retrieving data and populating it into the dHTMLx grid is fairly straight-forward.

The primary area of concern is how to then format the data from the native MII format into something that the dHTMLx grid can understand. This functionality is built into every MII Query Template via the ability to assign an XSLT transformation in the Query Template definition. Once the transform is created and defined for the query you can test its operation in XML test mode only.

© 2010 SAP AG

Usage

There are two XSL transforms defined in Appendix A of this document one is for simple XML format conversion and the other does numeric and date formatting and is slightly more complicated. However the principal behind both of them is the same and that is to format the SAP MII Illuminator Document XML into dHTMLx XML.

Simply import the attached project content into your SAP MII environment the project is named "ThirdPartyTools". The primary page for demonstrating the SAP MII integration is: ThirdPartyTools/WEB/dhtmlxSuite/Grid/samples/mii_integration/miidata_grid.html

How it Works

Basic Principles

The way that this integration works is by modifying the XML structure returned from the MII Query template into something that the dHTMLx grid can understand. This is done by assigning the following XLST from the project content to the inline transform property of the query template:

WEB://ThirdPartyTools/Common/XSLT/ConvertMIIXMLTodHTMLxFormat.xsl

OR

WEB://ThirdPartyTools/Common/XSLT/ConvertMIIXMLTodHTMLxFormat_AdvancedRowFormatting.xsl

Depending on what you are trying to accomplish either of these XSLT's will work for you. The ConvertMIIXMLTodHTMLxFormat.xsl file can be used for simple conversion of the SAP MII XML content to put raw data into the dHTMLx Grid and will perform faster than the ConvertMIIXMLTodHTMLxFormat_AdvancedRowFormatting.xsl will. However, the ConvertMIIXMLTodHTMLxFormat_AdvancedRowFormatting.xsl XSLT will allow more control over the format of the number and date formats which may be desired as times so this example is provided as well.

The dHTMLx grid supports a variety of built-in REST methods to help process data into the grid from various sources in a generic fashion. The JavaScript functions used to retrieve the data out of the MII application are as follows:

```
var mygrid = new dhtmlXGridObject('gridbox');
…
mygrid.loadXML("/XMII/Illuminator?QueryTemplate=ThirdPartyTools/dhtmlxSuite/Grid/samples/mii_integration/AssetUtilizationTagQuery_WithAdvRowFormatting&Content-Type=text/xml");
```

Things to note about the above URL is that for one it's a relative URL call, meaning that this web page is hosted on the MII server which mean you have already logged into MII to access the page. Therefore there's no need to authenticate this URL request. Secondly, the Content-Type value is defined as text/xml so that the raw XML data is returned to the dHTMLx grid. Finally, only the Query Template path needs to be specified however you can parameterize this query with the addition of URL parameters for Param.1 through Param.32. An example of this would be:

```
var myTag1 = "L1Speed";
mygrid.loadXML("/XMII/Illuminator?QueryTemplate=ThirdPartyTools/dhtmlxSuite/Grid/samples/mii_integration/AssetUtilizationTagQuery_WithAdvRowFormatting&TagName.1="+myTag1+"&Content-Type=text/xml");
```

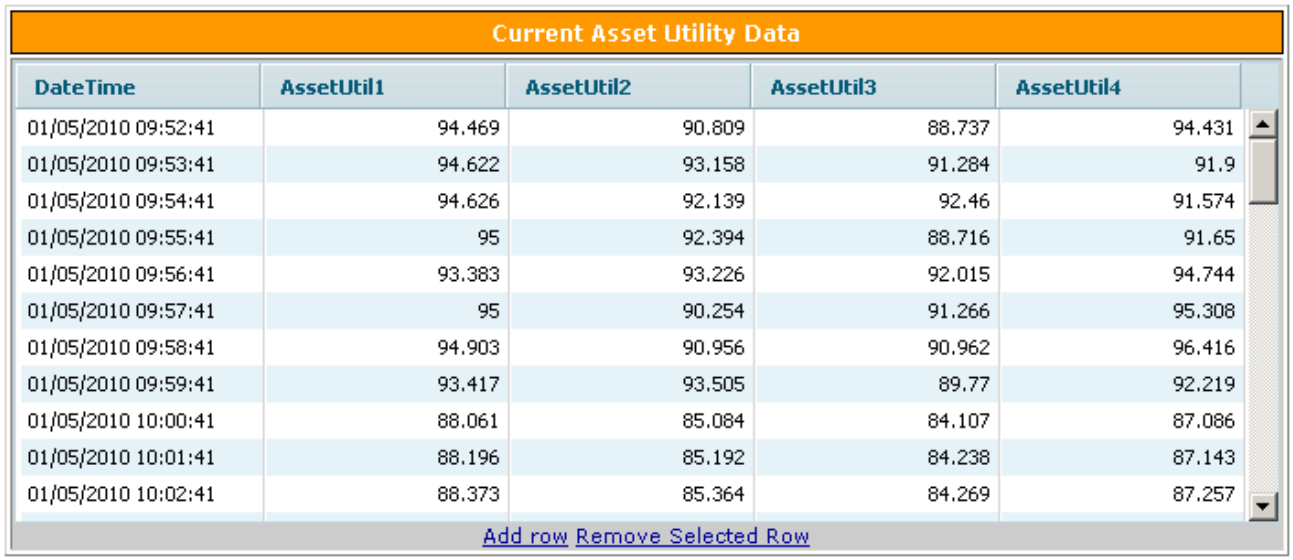
Advanced Features

It's also possible to define an ID column within the dHTMLx grid object and this requires the use of the ConvertMIIXMLTodHTMLxFormat_AdvancedRowFormatting.xsl transform in order for it to work properly. The name of the column that is returned in the query needs to be mapped to the IDColumn parameters of the XSLT in order for this to work properly.

© 2010 SAP AG

Result

The included example page, located here in the example project ThirdPartyTools/WEB/dhtmlxSuite/Grid/samples/mii_integration/miidata_grid.html, will create a simple editable grid page that looks like this in order to give you a working example to build from:



Current Asset Utility Data

DateTime	AssetUtil1	AssetUtil2	AssetUtil3	AssetUtil4
01/05/2010 09:52:41	94.469	90.809	88.737	94.431
01/05/2010 09:53:41	94.622	93.158	91.284	91.9
01/05/2010 09:54:41	94.626	92.139	92.46	91.574
01/05/2010 09:55:41	95	92.394	88.716	91.65
01/05/2010 09:56:41	93.383	93.226	92.015	94.744
01/05/2010 09:57:41	95	90.254	91.266	95.308
01/05/2010 09:58:41	94.903	90.956	90.962	96.416
01/05/2010 09:59:41	93.417	93.505	89.77	92.219
01/05/2010 10:00:41	88.061	85.084	84.107	87.086
01/05/2010 10:01:41	88.196	85.192	84.238	87.143
01/05/2010 10:02:41	88.373	85.364	84.269	87.257

Add row Remove Selected Row

This is just a simple example of what the editable grid can do, the example page does allow for the user to capture various editing events in the JavaScript such as cell enter, exit, previous and new values. This type of control allows for granular handling of user interaction and allows for a variety of requirements to be met. Try un-commenting various lines in the page definition to see the various trace and alert statements and also to modify the way the grid functions.

© 2010 SAP AG

Appendix A

ConvertMIIXMLTodHTMLxFormat.xsl

```
<!--
Use this transform to convert SAP MII XML into an XML format that the dHTMLx Grid can
understand for more information on the Grid see: http://dhtmlx.com/docs/products/dhtmlxGrid/

Author: Salvatore Castro
Date: July 17, 2009
-->
<xsl:stylesheet xmlns:java="http://xml.apache.org/xslt/java"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" exclude-result-prefixes="java"
version="1.0">
   <xsl:output method="xml" media-type="text/xml" encoding="UTF-8"/>
    <xsl:template match="/">
        <xsl:param name="DateFormat">MM/dd/yyyy HH:mm:ss</xsl:param> <!-- Default
pattern for formatting XML dates -->
        <rowsets>
            <rowset>
                <head>
               <xsl:for-each select="Rowsets/Rowset/Columns/Column">
         <xsl:call-template name="buildcol">
<!--     Here is an example of an MII XML Column defintion for reference on what is
being passed to the template -->
<!--             <Column Description="DateTime" MaxRange="1" MinRange="0"
Name="DateTime" SQLDataType="93" SourceColumn="DateTime" /> -->
                <xsl:with-param name="Description"><xsl:value-of
select="@DateTime"/></xsl:with-param>
                <xsl:with-param name="MaxRange"><xsl:value-of
select="@MaxRange"/></xsl:with-param>
                <xsl:with-param name="MinRange"><xsl:value-of
select="@MinRange"/></xsl:with-param>
                <xsl:with-param name="Name"><xsl:value-of select="@Name"/></xsl:with-
param>
                <xsl:with-param name="SQLDataType"><xsl:value-of
select="@SQLDataType"/></xsl:with-param>
                <xsl:with-param name="SourceColumn"><xsl:value-of
select="@SourceColumn"/></xsl:with-param>
                <xsl:with-param name="DateFormat"><xsl:value-of
select="$DateFormat"/></xsl:with-param>
         </xsl:call-template>
                </xsl:for-each>
                </head>
         <!-- From the MII XML simply copy the row information and it's child nodes
to the result XML document -->
         <xsl:copy-of select="/Rowsets/Rowset/Row" />
            </rowset>
        </rowsets>
    </xsl:template>

    <xsl:template name="buildcol" match="*">
   <xsl:param name="Description" />
   <xsl:param name="MaxRange" />
   <xsl:param name="MinRange" />
```


© 2010 SAP AG

```
   <xsl:param name="Name" />
   <xsl:param name="SQLDataType" />
   <xsl:param name="SourceColumn" />
   <xsl:param name="DateFormat" />
                  <column>
         <xsl:choose>
               <!-- Types:                                  91 = Date
92 = Time                        93 = DateTime -->
               <xsl:when test="$SQLDataType= '91' or $SQLDataType= '92' or
$SQLDataType= '93'">
                     <xsl:attribute name="align">left</xsl:attribute>
                     <xsl:attribute name="type">dhxCalendarA</xsl:attribute>
                     <xsl:attribute name="sort">str</xsl:attribute>
                     <xsl:attribute name="width">*</xsl:attribute>
                     <xsl:value-of
select="java:com.sap.xmii.Illuminator.ext.ExtFunctions.dateFromXMLFormat(string(.),$D
ateFormat)"/>
               </xsl:when>
                        <!-- Types:                          2 = Numeric
3 = Decimal                   4 = Long Integer          5 = Short Integer
6 = Float                        7 = Real                          8 = Double
-7 = Bit-->
               <xsl:when test="$SQLDataType = '2' or $SQLDataType = '3' or
$SQLDataType = '4' or $SQLDataType = '5' or $SQLDataType = '6' or $SQLDataType = '7'
or $SQLDataType = '8' or $SQLDataType = '-7'">
                     <xsl:attribute name="align">right</xsl:attribute>
                     <xsl:attribute name="type">ed</xsl:attribute>
                     <xsl:attribute name="sort">str</xsl:attribute>
                     <xsl:attribute name="width">*</xsl:attribute>
               </xsl:when>
               <!-- If the type is anything else just assign it basic string
formatting-->
               <xsl:otherwise>
                     <xsl:attribute name="align">left</xsl:attribute>
                     <xsl:attribute name="type">ed</xsl:attribute>
                     <xsl:attribute name="sort">str</xsl:attribute>
                     <xsl:attribute name="width">*</xsl:attribute>
               </xsl:otherwise>
         </xsl:choose>
         <xsl:value-of select="$Name" />
                  </column>
    </xsl:template>
</xsl:stylesheet>
```


© 2010 SAP AG

ConvertMIIXMLTodHTMLxFormat_AdvancedRowFormatting.xsl

```
<!--
Use this transform to convert SAP MII XML into an XML format that the dDHTMLx Grid
can understand for more information on the Grid see:
http://dhtmlx.com/docs/products/dhtmlxGrid/

Author: Salvatore Castro
Date: July 17, 2009
-->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:java="http://xml.apache.org/xslt/java" exclude-result-prefixes="java">
   <xsl:output method="xml" media-type="text/xml" encoding="UTF-8"/>
    <xsl:template match="/">
        <xsl:param name="DateFormat">MM/dd/yyyy HH:mm:ss</xsl:param> <!-- Default
pattern for formatting XML dates -->
        <xsl:param name="NumberFormat"></xsl:param> <!-- Default pattern for
formatting XML dates -->
        <xsl:param name="IDColumn"></xsl:param> <!-- Column that contains the IDs for
the result set -->
        <rowsets>
            <rowset>
   <xsl:for-each select="Rowsets/Rowset">
          <xsl:variable name="CurrentColumns" select="Columns"/>
          <head>
                  <xsl:for-each select="Columns/Column">
                <xsl:call-template name="buildcol">
<!--            Here is an example of an MII XML Column defintion for reference on
what is being passed to the template -->
<!--                    <Column Description="DateTime" MaxRange="1" MinRange="0"
Name="DateTime" SQLDataType="93" SourceColumn="DateTime" /> -->
                      <xsl:with-param name="Description"><xsl:value-of
select="@DateTime"/></xsl:with-param>
                      <xsl:with-param name="MaxRange"><xsl:value-of
select="@MaxRange"/></xsl:with-param>
                      <xsl:with-param name="MinRange"><xsl:value-of
select="@MinRange"/></xsl:with-param>
                      <xsl:with-param name="Name"><xsl:value-of
select="@Name"/></xsl:with-param>
                      <xsl:with-param name="SQLDataType"><xsl:value-of
select="@SQLDataType"/></xsl:with-param>
                      <xsl:with-param name="SourceColumn"><xsl:value-of
select="@SourceColumn"/></xsl:with-param>
                      <xsl:with-param name="DateFormat"><xsl:value-of
select="$DateFormat"/></xsl:with-param>
                </xsl:call-template>
                  </xsl:for-each> <!-- Columns/Column -->
          </head>
                  <xsl:for-each select="Row">
                       <Row>
                             <xsl:choose>
                                        <!-- If an ID Column is not defined -->
                                  <xsl:when test="$IDColumn= '' ">
                                        <xsl:attribute name="id"><xsl:value-of
select="position()"/></xsl:attribute>
```


© 2010 SAP AG

```
                                    </xsl:when>
                                          <!-- Else an ID Column is defined -->
                                    <xsl:otherwise>
                                          <xsl:attribute name="id"><xsl:value-of
select="./*[name() = $IDColumn]"/></xsl:attribute>
                                    </xsl:otherwise>
                              </xsl:choose>

                        <xsl:for-each select="*">
                        <xsl:variable name="CurrentNodeName" select="name(.)"/>
                        <xsl:call-template name="buildrow">
                              <xsl:with-param name="DateFormat"><xsl:value-of
select="$DateFormat"/></xsl:with-param>
                              <xsl:with-param name="NumberFormat"><xsl:value-of
select="$NumberFormat"/></xsl:with-param>
                              <xsl:with-param name="NodeValue"><xsl:value-of
select="."/></xsl:with-param>
                              <xsl:with-param name="SQLDataType"><xsl:value-of
select="$CurrentColumns/Column[@Name=$CurrentNodeName]/@SQLDataType"/></xsl:with-
param>
                        </xsl:call-template>
                        </xsl:for-each> <!-- Each cell in the rows -->
                        </Row>
                  </xsl:for-each> <!-- Row -->
   </xsl:for-each> <!-- Rowsets/Rowset -->
            </rowset>
         </rowsets>
    </xsl:template>

    <xsl:template name="buildrow" match="*">
   <xsl:param name="SQLDataType" />
   <xsl:param name="NodeValue" />
   <xsl:param name="DateFormat" />
   <xsl:param name="NumberFormat" />
   <cell>
   <xsl:choose>
                  <!-- Types:                     91 = Date                        92 =
Time                        93 = DateTime -->
          <xsl:when test="$SQLDataType= '91' or $SQLDataType= '92' or $SQLDataType=
'93'">
                  <xsl:value-of
select="java:com.sap.xmii.Illuminator.ext.ExtFunctions.dateFromXMLFormat(string($Node
Value),$DateFormat)"/>
          </xsl:when>
                  <!-- Types:                        2 = Numeric                   3 =
Decimal                   4 = Long Integer          5 = Short Integer          6 =
Float                        7 = Real                          8 = Double
-7 = Bit-->
          <xsl:when test="$SQLDataType = '2' or $SQLDataType = '3' or $SQLDataType =
'4' or $SQLDataType = '5' or $SQLDataType = '6' or $SQLDataType = '7' or $SQLDataType
= '8' or $SQLDataType = '-7'">
                  <xsl:value-of select="format-number($NodeValue,$NumberFormat)"/>
          </xsl:when>
          <!-- If the type is anything else just assign it basic string formatting,
which in this case means not to do anything to the value -->
          <xsl:otherwise>
```


© 2010 SAP AG

```
                <xsl:value-of select="$NodeValue"/>
          </xsl:otherwise>
    </xsl:choose>
    </cell>
     </xsl:template>

     <xsl:template name="buildcol" match="*">
    <xsl:param name="Description" />
    <xsl:param name="MaxRange" />
    <xsl:param name="MinRange" />
    <xsl:param name="Name" />
    <xsl:param name="SQLDataType" />
    <xsl:param name="SourceColumn" />
    <xsl:param name="DateFormat" />
                    <column>
          <xsl:choose>
                <!-- Types:                                  91 = Date
92 = Time                        93 = DateTime -->
                <xsl:when test="$SQLDataType= '91' or $SQLDataType= '92' or
$SQLDataType= '93'">
                      <xsl:attribute name="align">left</xsl:attribute>
                      <xsl:attribute name="type">dhxCalendarA</xsl:attribute>
                      <xsl:attribute name="sort">str</xsl:attribute>
                      <xsl:attribute name="width">*</xsl:attribute>
                </xsl:when>
                        <!-- Types:                          2 = Numeric
3 = Decimal                   4 = Long Integer          5 = Short Integer
6 = Float                         7 = Real                          8 = Double
-7 = Bit-->
                <xsl:when test="$SQLDataType = '2' or $SQLDataType = '3' or
$SQLDataType = '4' or $SQLDataType = '5' or $SQLDataType = '6' or $SQLDataType = '7'
or $SQLDataType = '8' or $SQLDataType = '-7'">
                      <xsl:attribute name="align">right</xsl:attribute>
                      <xsl:attribute name="type">ed</xsl:attribute>
                      <xsl:attribute name="sort">str</xsl:attribute>
                      <xsl:attribute name="width">*</xsl:attribute>
                </xsl:when>
                <!-- If the type is anything else just assign it basic string
formatting-->
                <xsl:otherwise>
                      <xsl:attribute name="align">left</xsl:attribute>
                      <xsl:attribute name="type">ed</xsl:attribute>
                      <xsl:attribute name="sort">str</xsl:attribute>
                      <xsl:attribute name="width">*</xsl:attribute>
                </xsl:otherwise>
          </xsl:choose>
          <xsl:value-of select="$Name" />
                    </column>
    </xsl:template>
</xsl:stylesheet>
```


© 2010 SAP AG

Related Content

The MII Example Project

The dHTMLx Reference Site for Additional Information: http://www.dhtmlx.com/docs/products/dhtmlxGrid/

(The sample content in this project is from this web site).

SAP MII Wiki: https://wiki.sdn.sap.com/wiki/display/xMII

SAP MII Forum: https://www.sdn.sap.com/irj/sdn/forum?forumID=237

© 2010 SAP AG

Copyright

© Copyright 2010 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects S.A. in the United States and in other countries. Business Objects is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.