SDN Contribution

SAP xMII Best Practices Guide

Applies to:

SAP xMII 11.0 to 11.5

Summary

Due to the varying functionality and flexibility of design with xMII, it is beneficial to have a best practices guide on using xMII, structuring the XHTML/HTML/IRPT and editing JavaScript & CSS in a standard manner. This guide will help to point out various approaches when using xMII to allow for the most flexibility and greater performance of your application.

Author(s): xMII Field Enablement and Support Teams

Company: SAP Labs, LLC

Created on: 11 January 2007

Table of Contents

Introduction

Due to the varying functionality and flexibility of design with SAP xMII, it is beneficial to have a best practices guide on using xMII, structuring the XHTML/HTML/IRPT content, as well as editing JavaScript and CSS in a standard manner. This document will show some high level techniques that should be followed in order to achieve a standardized use of the product and for future troubleshooting and organization.

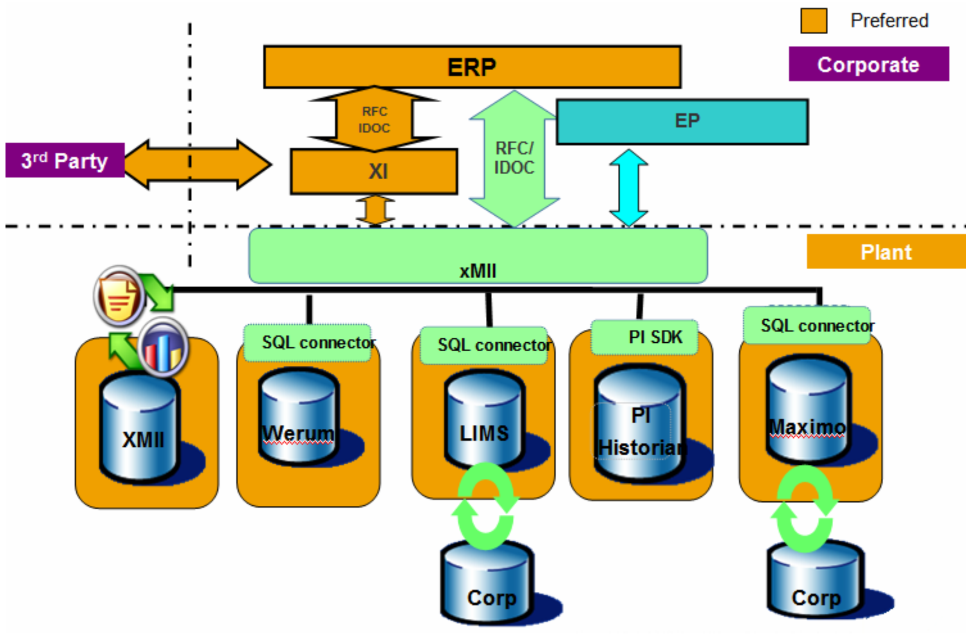
1 Architecture

1.1 xMII Integration Architecture

SAP xApp Manufacturing Integration and Intelligence (xMII) integrates with manufacturing systems and mySAP ERP Central Component or R/3 to connect the operation level and enterprise data. This allows you to monitor your operations and resolve manufacturing exceptions in real time.

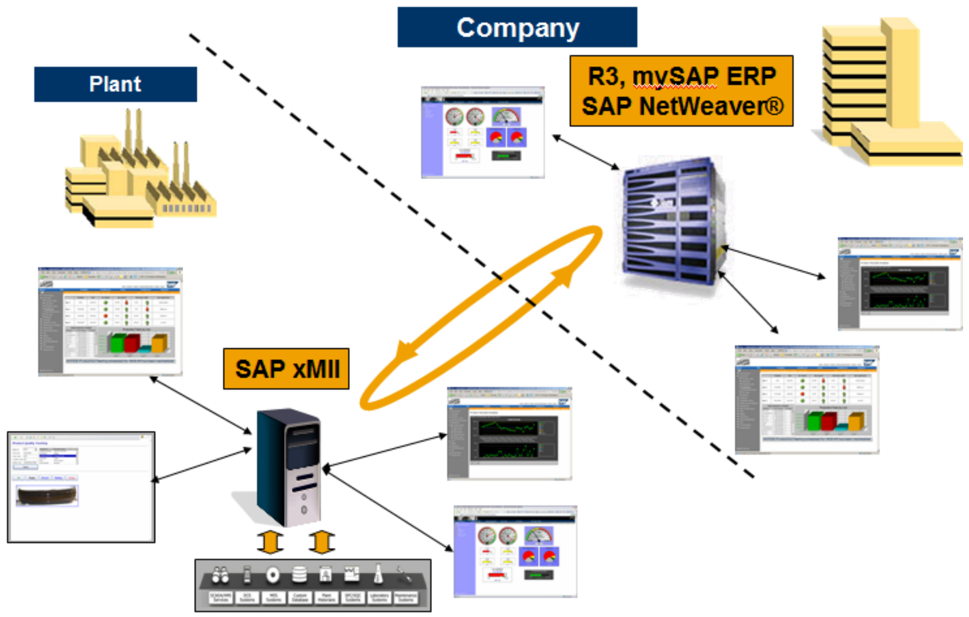
In a NetWeaver environment, xMII offers comprehensive "pre-integration" to virtually all elements of SAP ERP and the NetWeaver technology stack. Content and services created in the xMII environment are exposed to the entire enterprise as services via NetWeaver. All xMII services can be utilized by other NetWeaver components (XI, Portal, and Visual Composer). Visualizations can be exposed through SAP Portal, and content delivery through Microsoft SharePoint is also supported.

Support access to 100% of ERP functionality integration through 3rd Party Middleware/EAI products as well.



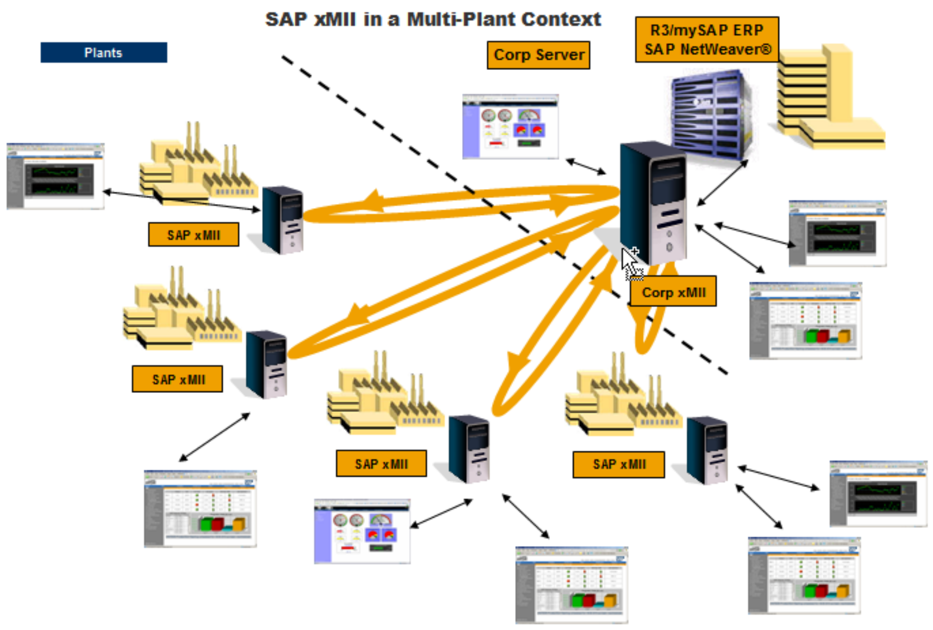
1.2 SAP xMII Single Plant/Region Architecture

In the single plant example below, a single site xMII server is installed on the plant network to provide operational integration and visibility for production personnel. One of the reasons for having xMII server at each plant is that most of the applications xMII interacts with and connects to, such as control systems and process historians, are operational and required at each plant. The data volume generated from some of these targets is very high and transferring it across the network may impact performance. In addition, having an xMII server at each plant allows you the option of running in a disconnected mode in case of wide area network (WAN) outage or ERP maintenance.



1.3 SAP xMII Enterprise Architecture

In the multi-plant example below, each plant site has an xMII server. An additional xMII corporate server can then be used to aggregate and compare multi-site data to provide plant-to-plant or divisional views and analytics for a complete Balanced Scorecard or best-of-class comparison.



2 Server Configuration Best Practices

2.1 Web Servers

It is recommended that each xMII project should have at least two servers; a Development and Production server; an optional QA instance is also preferred. The two server (Dev. & Prod) configuration and installations should be identical for fast and easy promotion and deployment from the development to the production environment.

2.2 Web Applications

2.2.1 Definitions

Web Application – a web based application that is developed using SAP xMII.

Web Server – the machine that is to host the xMII software; Microsoft IIS

Web Application Server – the back-end servlet engine used to supplement the functions of a web application; New Atlanta's ServletExec

<root> - the installation disk drive letter such as C:, D:, etc.

2.2.2 Deployment Strategies

Each web server should have a separate web site configured in IIS that will be used for the installation of xMII. If the web server has only one web site configured, then the Default Web Site can be used for ease of configuration and administration. The ServletExec Java Servlet engine will be installed for that specific web site, and the xMII product runs within ServletExec. All requests to the web site are redirected to ServletExec for handling.

Individual developed application directories should be installed below the root of the web site. Corresponding xMII template directories, with the same names, should be created for each web server, under the templates directory. xMII should be installed on the drive of the machine where IIS is installed. A Business Logic transaction directory with the same name should also be created for each web server under the transactions directory.

In general for each web server you should have the following directories with corresponding names

<root>\Inetpub\wwwroot\<CompanyName>

<root>\Lighthammer\Illuminator\Templates\<CompanyName>

<root>\Lighthammer\Xacute\Transactions\<CompanyName>

In addition to the three base company name directories, there should be a common directory located underneath each location for common, reusable objects. Using common objects promotes team development and standardization, helps maintains application consistency, and provides project team members with reusable content.

Examples of common objects are as follows:

- Common queries across many applications (e.g. - an equipment list query)
- Common user input objects, such as date selection dialogs
- Common graphics and other images
- Standard display objects, such as drop down iBrowser objects for web forms
- Common Business Logic transactions across application (file I/O, e-mail messaging, etc.)
- Reusable JavaScript library files
- Cascading Style Sheets (CSS)

A web site administrator or xMII lead architect should control the content of common directories. There should be a submittal and approval process before common objects are available to all application developers. The web administrator should also control the modification and maintenance of the configuration files found in the following directory locations:

<root>\Lighthammer\Security\sysconf\

<root>\Lighthammer\Security\sysconf\userdata

<root>\Lighthammer\Illuminator\Conf

<root>\Lighthammer\Illuminator\SysConf

The above paths contain files which control servers, time periods, users, roles, and security through the use of their associated administration and configuration screens. These files are not to be edited by hand unless directed to do so by an experienced technical support representative.

2.3 Installation & Migration

Installation Recommendations

The recommendations in this section require careful consideration, and should not be ignored without first understanding the implications. The standard xMII "Classic" theme (black background) should be selected during install in order to keep the development and any future support consistent with legacy installations. Content examples provided through SDN will be developed using this theme as a basis for development.

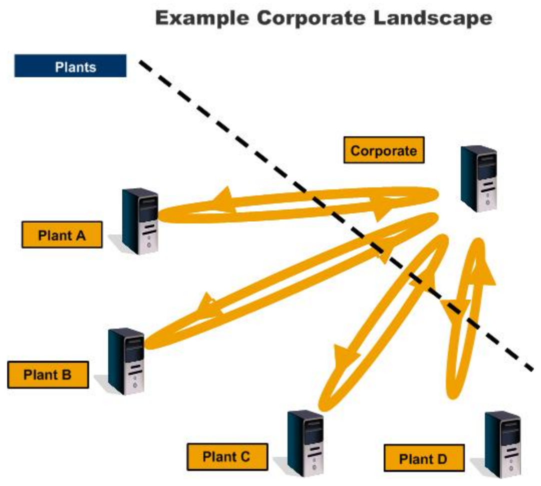
Recommendations

The installation and upgrade procedures guide in the help documentation (http://help.sap.com under xApps -> SAP xApp Manufacturing Integration and Intelligence) should be reviewed and fully understood BEFORE anything is done.

Each instance of SAP xMII should have at least a development and a production instance. It is also a recommendation to have a full backup of your development and production instances made before the upgrade is performed.

The development instance should be upgraded and fully tested prior to the production instance.

The upgrade should begin with the corporate instance and then rollout to each of the plants.



Information about all customization should be recorded. This includes all custom action blocks, customized style sheets, customized themes, custom SVA/SVG images, custom reference documents, and any other customization done on the web root, Illuminator, or in the Xacute directories other than Templates and Business Logic transactions.

Please be aware that custom action blocks written in .NET will not work with the 11.5.x and newer versions. The 11.5.x version of business logic is Java based and will require you to convert your .NET code to Java before your action blocks will work.

Typical directories for customization are, but not limited to, the following locations:

<root>\Inetpub\wwwroot\Illuminator\*

<root>\Lighthammer\Illuminator\Templates\*

<root>\Lighthammer\Illuminator\Properties\*

<root>\Lighthammer\Illuminator\Conf

<root>\Lighthammer\Illuminator\SysConf

<root>\Lighthammer\Xacute\AnimatedObjects\*

<root>\Lighthammer\Xacute\Components\*

<root>\Lighthammer\Xacute\Globals\*

<root>\Lighthammer\Xacute\ReferenceDocuments\*

<root>\Lighthammer\Xacute\SVGObjects\*

<root>\Lighthammer\Xacute\Transactions\*

Performing the Upgrade (How to Upgrade Lighthammer Illuminator)

Begin with the development instance of your corporate server.

Follow the upgrade procedures specified in the product help documentation on http://help.sap.com under xApps -> SAP xApp Manufacturing Integration and Intelligence.

The development server should be thoroughly tested before the upgrade is performed on the production instance. There is a backup configuration stored on the installation drive of the Illuminator server. The recommended time for this testing period will vary depending upon the developed application, but at a minimum should include a complete click through of the entire application and its functionality.

There are also logs stored on the server indicating the status of the migration. These should be checked if you run into any issues before support is contacted.

The automatic backup performed by the upgrade is located in the following directory:

<root>\Illuminator_backup_Date_Time

The upgrade log files are located in:

<root>\Lighthammer\Logs

Migration.log, CMSSecurityMigration.log, and SAP_xMII_Install_Date_Time.log

* If you begin to have issues with your SAP xMII server instance please consult these logs first before contacting support.

Once the development server has been upgraded and validated repeat the procedure for the production environment. This should be done per your company's production server upgrade procedures as far as notification of system downtime and scheduling as to not interfere with your production.

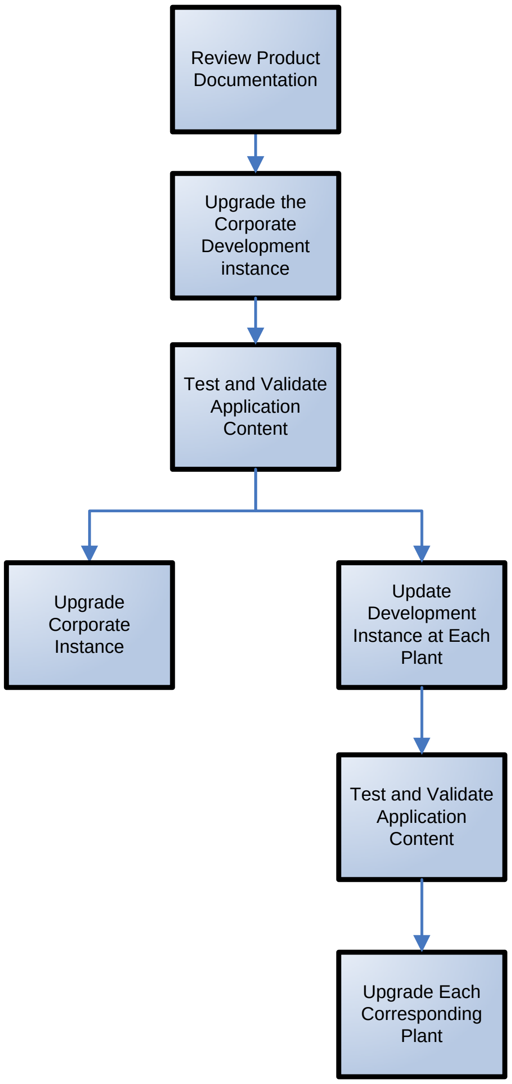
While the production server upgrade is taking place it is ok to move ahead with upgrading the development instances at each of your plants.

Each of these instances should go through a similar validation process that was performed on the corporate instance.

Once the operation of the server is validated it is ok to upgrade the corresponding production instance.

If virtual server connections were used for server to server connectivity between corporate and plant instances note the LegacyURL check box in the Data Server configuration. When the legacy box is checked then a connection to an older version of the product can be made.

Below is a procedural flowchart of the upgrade process:


Review Product Documentation
Upgrade the Corporate Development instance
Test and Validate Application Content
Upgrade Corporate Instance
Update Development Instance at Each Plant
Test and Validate Application Content
Upgrade Each Corresponding Plant

Migration Failure Recovery Recommendations

Please note that the backup configuration xml files are still in 10.x format and cannot simply be copied into the 11.x environment. If your migration does fail then revert back to the 10.x installation and copy your backup xml files over your 10.1 reinstallation and retry your 11.x migration. Be sure that when you reinstall 10.1.x that you do not copy over the backed-up configuration files.

Caveats for Future Release Compatibility

ASP pages will not allow for multi-platform capabilities found inherent with the Web AS architecture in version 12.0. A proposed work around for this would be to have an IIS server that the page could post to that would support the ASP capabilities.

3 Application Design Best Practices

Standard web page templates should be created and used as a starting point for all new web page development. Refer to the sample page syntax below:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<title>PAGE TITLE HERE</title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<meta http-equiv="Expires" content="0">

<meta http-equiv="Cache-Control" content="no-cache">

<meta http-equiv="Pragma" content="no-cache">

<link rel="stylesheet" type="text/css" href="/CompanyName/Common/CompanyName.css">

<script type="text/javascript" src="/CompanyName /Common/UtilFunctions.js"/>

</head>


<body>

        <!-- PAGE CONTENT HERE -->

</body>

</html>
```

Each web page should be interpreted as three separate components:

Structure component defines the initial objects of a web page (.htm, .irpt, .jsp, etc.).

Presentation component defines the visual aspect of a web page via cascading styles applied to HTML objects. All styles should reside in an external cascading stylesheet library (.css). To maintain consistency, web pages should contain references to common Cascading Style Sheets.

Behavior component defines the dynamic interactivity and creation of objects in a web page, which should be defined in an external JavaScript library (.js). Where appropriate, create common libraries relative to specific functionality of the web application.

***Note: The standard xMII "Classic" theme (black background) should be selected during install in order to keep the development and any future support consistent with legacy installations. Content examples provided through SDN will be developed using this theme as a basis for development.

3.1 Structure DOs & DON'Ts

DO/DON'T	Task	Why?
DO	Define a document type definition (DTD) using the ***W3C HTML 4.01 Transitional*** standard	***HTML 4.01 Transitional DTD*** is the standard that supports APPLET tags
DO	Define all JavaScript blocks using <script t*ype="text/javascript"></script>*	W3C has deprecated the usage of <script language*="javascript"></script>*

DO/DON'T	Task	Why?
DO	Use ***<servlet>*** tags for displaying data instead of an applet in a static, printable format when user events are not needed.	Servlets are recommended for use with non-Java (Standard Edition) mobile devices.
DO	Drive multiple applet loading from xMII applet events (Selection events, button click event) rather than synchronizing via <body onload="myFunction();"> or from the loading of another applet.	Java applets are embedded objects, which are entirely independent of the document. Therefore, it is not possible to predict applet behavior or the order in which they load on a page.
DO	Evaluate each xMII Applet definition; Take advantage of the supplied xMII Productivity Wizards to assist in the applet code generation.	A high occurrence of user-error has been identified within illegal applet definitions, typically caused my manual editing.
DON'T	Do not place applets in a <div> tag with height and width as a percentage; rather define the height and width as pixels or other measurement.	When defined as a percentage, the browser windows size changes the div size which could interfere with the applet rendering correctly.

3.2 Presentation DOs & DON'Ts

A common cascading stylesheet library should be used to keep the same look and feel between web pages within the xMII web application. If you are using FrontPage, all text fonts and sizes should be selected from the Style dropdown, and not by directly choosing font, size, and weight. For a project starting point, the standard default stylesheet (<root>\Inetpub\wwwroot\Illuminator\StyleSheets\cms.css) can be copied to the Common project directory underneath the webroot and should be managed by the web administrator or lead architect.

DO/DON'T	Task	Why?
DON'T	Attempt to manipulate applets via CSS	Although applets can be displayed inside HTML, applets are rendered independently of HTML.
DON'T	Use inline CSS within a web page; instead use an external style sheet or one defined within the web page header	Constraining styles to a global, reusable style sheet allows a streamlined approach to implementing and if necessary troubleshooting CSS.

3.3 Behavior DOs & DON'Ts

DO/DON'T	Task	Why?
DO	All variables need to be declared; it is good practice to declare them at the top of a function unless where necessary otherwise. Avoid using key or reserved words when naming the variables.	Most scripting languages require that all variables be declared. Even though JavaScript does not require this compliancy, consistency in programming eases future development.
DO	Provide proper documenting and commenting of all relative code.	Commenting code eases future development and promotes team level reusability of functionality.

DO	Consider the creation of JavaScript variables to reference the Applet object, Display Object, and Query Object.	var myApplet = document.iGrid;
		var myGrid = myApplet.getGridObject();
		var myQuery =
		myApplet.getQueryObject();

DO	Use the xMII Script Assistant to generate and determine correct query/display object methods and properties.	The xMII Script Assistant is the premier tool to reference applet methods. The standalone version can be used as a quick reference to ensure adherence to the proper JavaScript syntax.
DON'T	Do not use JavaScript functions dealing with applets within the <body> of a web document; instead define them in a <script/> block located in the <head/> of a web page or in a separate js file.	W3C recommends that the <script> tag be located between the <head> tags.
DON'T	Do not use ASP, VBScript, or Jscript to interact with the applets.	JavaScript is the only documented and supported scripting language for interfacing with the applets.

3.4 Application Directory Structure

A web folder should be created for each functional area of an xMII web application to provide organization and structure to the application itself.

Example:

> <root>\Inetpub\wwwroot\CompanyName\AppName1\FunctionalSet1
>
> <root>\Inetpub\wwwroot\CompanyName\AppName1\FunctionalSet2
>
> <root>\Inetpub\wwwroot\CompanyName\AppName2\FunctionalSet1
>
> <root>\Inetpub\wwwroot\CompanyName\AppName2\FunctionalSet2
>
> <root>\Inetpub\wwwroot\CompanyName\AppName2\FunctionalSet3

3.5 Default Templates

xMII provides a set of default templates that are used as a foundation for controlling the typical look and feel of the applet visual properties. These default templates are located in <root>\Lighthammer\Illuminator\Templates\Defaults. When using the Display Template Editor and creating a New template, these default templates assign the color, string, and boolean properties to the various configuration parameters available for the corresponding iChart, iGrid, etc. Do not make changes to the default system templates located in the Defaults template folder. Note: the iGrid is auto-defaulted to the xMII default template (/Defaults/iGrid). In addition to those provided the display templates similar to the following list should be created and used to create the appropriate applet types such as:

> <root>\Lighthammer\Illuminator\Templates\CompanyName\DefaultTheme\iChart
>
> <root>\Lighthammer \Illuminator\Templates\CompanyName\DefaultTheme\iGrid
>
> <root>\Lighthammer \Illuminator\Templates\CompanyName\DefaultTheme\iSPCChart

These template themes will insure common colors, font style and sizes, and properties for all newly developed content. Once the company desired default theme templates have been created, the web administrator should make the xml files Read-Only to preserve the standards and to prevent inadvertent developer Save (instead of Save As) errors. When a new template is needed, application team members should load the appropriate default theme template, and use the Save As function to save the template to the desired application directory, where content specific changes can then be made. The Writer Role security can also be adjusted so that only the web administrator role has the ability to change the Default Theme templates.

3.6 Naming Convention Suggestions

All query templates should be named in the following manner:

<SpecificPurpose>Query

For example a Shift Detail Query template would be named:

ShiftDetailQuery, where Shift Detail is the specific implementation, and Query identifies the object as a query object. Keep in mind that template names are always provided as fully qualified paths, so the directory structure itself carries additional information that does not need to be reiterated in the specific template name itself. For example, if the ShiftDetailQuery template was located in a project path such as CompanyABC/Metrics/OEE/, the simple name could remain ShiftDetailQuery, but when including the full path of the template would read CompanyABC/Metrics/OEE/ShiftDetailQuery and would classify the ShiftDetailQuery as a component of the OEE Metric for CompanyABC.

For iGrids, iCharts, iBrowser, SPC Charts and Ticker display templates, substitute the appropriate object type for "Query" in the template name.

4 General xMII Best Practices

The following points are general xMII functionality suggestions that should be used in order to achieve the full potential of the product. These points will also be helpful in fine tuning the performance of xMII for maximum usability and response time.

4.1 Business Logic

DO/DON'T	Task	Why?
DO	Document action blocks used in Business Logic and their respective segments with a meaningful description and name.	Makes it easier for users to understand, troubleshoot and rework a transaction.
DO	For complex transactions, it is helpful to have an empty segment at the beginning of each branch with a detailed description of what that part of the transaction does.	Quick glance gives user an idea of what the following actions will be accomplishing.
DO	Avoid unnecessary that are used for debugging.	Disk I/O is expensive and using a Tracer can be much easier to follow.
DO	Use a transactional Boolean property such as 'DebugFlag' in a Conditional action block in the sequence immediately before doing the XMLSavers to allow for easy debugging.	Allows for dynamic interaction of debugging user selection. Saves on processing time by not running debug actions every time a transaction is queried.
DO	Use error handling messages when dealing with SAP data sources and other sources as well.	Helps in troubleshooting data access problems easily and efficiently
DO	Reference arrows on action blocks for a visual of incoming and outgoing parameter assignments	Easy method of quickly determining an action's configuration
DON'T	Replicate action blocks under the True/False branches of a conditional. Instead, declare local variables to hold the values calculated under each condition. Add a third branch in which the necessary action blocks can call the reference variables.	Simplifies the logic in a transaction; also improves processing time when querying a transaction
DON'T	Declare date parameters as Strings instead of the native DateTime format.	Will make date-time calculations much easier and faster.

Optimizing Business Logic Transactions with XPath – reference on SAP Developer Network showing optimization techniques for XML manipulation using XPath

4.2 Querying & Caching

Query caching in the xMII environment, when used properly, can greatly improve the efficiency of your xMII application. For introductory knowledge of the xMII cache please read in the information contained in the xMII help docs under DataServices -> Query Templates -> Query Editor Overview -> Query Editor.

The two main things to keep in mind when setting the query cache duration is the volatility of the data stored in the data source and the resolution that the user requires the data to be viewed at.

Since the level of volatility is not consistent for all values across a system it is important to know the business process in place in order to accurately set your cache.

For example look at two hypothetical tags that exist in a Plant Information (PI) system:

Tag_A, represents a count of bottles through the production line. It represents a value that is constantly changing from second to second.

Tag_B, represents the current batch. It may only change every couple of hours.

For the case of Tag_A let's say that there is a page that is displaying the total number of bottles produced versus the total number of bottles rejected and then their difference as shown below:

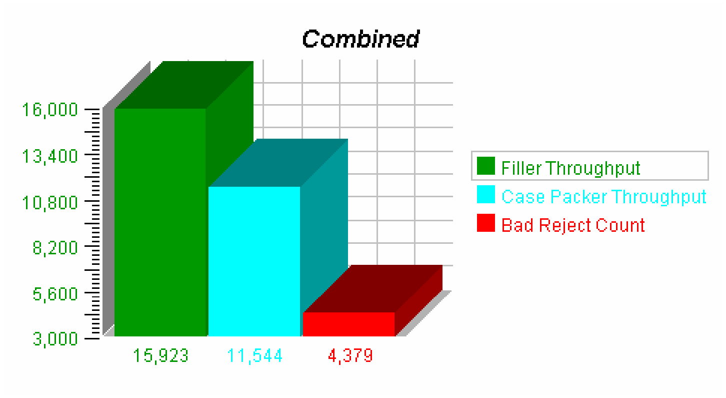


Figure 1: Bar chart of piece counts at various stages on the production line.

There is typically no need to query the PI system every time that this applet is loaded on a page for up to the second counts. Rather the query can be cached for a couple of minutes to save the PI system from handling a lot of queries and it will still yield enough information about the process to be useful. For the Tag_B type information about current batch, depending on your business process, it may be ok to cache the results for half an hour. Worst case scenario for this would be data coming back that was 59 minutes old but in the case of current batch this may be alright for the front end. However, if the query is for some back end system you may not wish to cache it at all.

A slightly different situation for using the xMII cache is where the data values change once a day or even once a week such as with Business Information Warehouse (BI or BW). In this situation the data coming back from the warehouse may take longer than the user cares to wait for their reports, leading to performance complaints and people becoming discouraged from using the interface. Additionally the need to query this data-source multiple times for the same data values will almost certainly arise. To circumvent this problem the cache duration can be set for multiple hours or even days before the information stored is set to expire. However there can be a problem with setting the cache for long time durations. Imagine the scenario where every morning at 8am the BW system has finished being loaded with the data from the previous day. The query cache on your query is set for 20 hours, in order to accommodate people in different time zones. If the query is made before noon that day there will not be a problem but if the query happens at 1pm and someone performs the query at 8:30am the following day they will still get information from the previous day. In order to prevent this situation from occurring it is possible to clear the xMII system cache via a URL servlet call to the xMII server: http://<ServerName>/Lighthammer/Illuminator?Service=QueryCaching&Mode=ClearCache

This will remove all of the cached values ensuring that there aren't any dataset values from the previous day. In addition to this servlet call it is possible to pre-charge the cache for the day by running a transaction that will make the "frequent" calls to the data source. An example of this situation would be a plant list from ERP. The list of plants will typically not change during the day or even during the week so it may be ok to query the system for this list of plants in the morning via a transaction and then store the results to the xMII cache. Then when the user, via the web page interface will not have to wait for this list of plants but rather will get them back instantly via the xMII cache.

Remember not to get discouraged with incorrect cache duration settings as this is specific to a business process and there's no concrete method for specifying them but rather is based off of the GUI requirements.

DO/DON'T	Task	Why?
DO	Query for data that is needed and not for full tables.	Minimizes load on the server and only displays what is necessary for the user to see.
DO	Enable query caching after the application has been developed.	Avoid issues during development; This will help to eliminate incorrect query issues due to the query results stored in the xMII cache.
DO	Use caching where the data values change once a day or even once a week.	Greatly improves response time of data as well as minimizing load on the server.
DON'T	Use date ranges when caching a query. Query caching will not work.	The reason behind this is to prevent large quantities of repetitive data from being stored in the xMII cache.

4.3 Session Variables

Session variables can be used to store specific data regarding the user and the current login session that is active. These can then be accessed through the IRPT pages created.

The default variables are: Description, FullName, Machine, IllumLoginName, IllumLoginRoles, and Language and can be viewed with http://<servername>/Lighthammer/PropertyAccessServlet?Mode=List

HTTP Request variables can also be passed into .IRPT pages via the URL, but these are only usable by that page because they are not in the actual session (i.e. http://<servername>/CompanyName/App1/mypage.irpt?OrderNum=123&RowNum=100).

DO/DON'T	Task	Why?
DO	Use the IRPT pages to access the Illuminator session variables.	Session variables are only available and accessible via the IRPT pages.
DO	Use the User Management Editor to create new session variables and assign values to them.	These new parameters would be available on user login.
DO	Use the setPropertyValue("NAME", "VALUE") method of an applet to set a session variable.	The method can be accessed via JavaScript when an applet is on a page. You can hide the applet by setting its height and width attributes to 1 so it's not viewable to a user.
DO	Set session variables via an applet's PARAM attributes.	Passing variables via PARAMs is useful when not using IRPT pages and only HTML since HTML can't handle it through the URL.
DO	Use the getPropertyValue("NAME") method of an applet to read a session variable.	Very easy method of extracting either preset or custom session variables.

5 Shop Floor Integration

Process Historians are widely used in manufacturing and other industries to record real-time process data from the shop floor, either utilizing a relational database or a combination of relational database and compressed archive files. Examples of these process data points, commonly referred to as tagnames, are as follows:

Analog: temperature, pressure, humidity, flow-rates, levels, weights

Digital/Discrete: limit switches, motors on/off, discrete level sensors

String: product id, batch id, material id, lot id

SAP xMII provides connectivity to a large number of historian software packages, connecting through a relational database interface, OleDB, or vendor specific API or SDK data connectors. These connectors provide not only the necessary means to interface with the underlying process historian, but they also provide the means to retrieve data in a consistent fashion, irregardless of the specific software package delivered by the vendor. All of the tags and respective data found in these tag based systems are exposed by the xMII connectors without requiring specialized knowledge about the underlying queries necessary to retrieve the desired information.

The xMII connectors dynamically expose the tags with namespace browsing modes such as:

GroupList: hierarchical tree structure for logical grouping of associated tagnames – not all historians implement this capability

TagList: list of tagnames and their corresponding description

The xMII connectors return data from the underlying historian with tag query modes such as:

Current: live tag value

History: time series interpolated tag values

HistoryEvent: time series raw tag values

Statistics: interval based statistical tag results from raw tag values

Please refer to the xMII documentation for additional information concerning query properties that are specific to 'TagQuery' query templates, as well as the connector specific documentation that identifies any vendor specific behavior or characteristics.

6 NetWeaver (BI, XI, EP, VC) Integration

How to Integrate xMII with the Business Information Warehouse – SAP Developer Network document highlighting the process for integrating xMII with BI

Calling Services and Queries in SAP xMII 11.5 from ABAP

How to Integrate SAP xApp Manufacturing Integration and Intelligence -SAP xMII with Visual Composer – SAP Developer Network document highlighting the process for integrating xMII with Visual Composer

How to Send an IDoc from the SAP R3 Enterprise to the SAP xMII IDOC Listener

7 Mobile Applications

How to configure the xMII Server and xMII Pages for Mobile Devices – SAP Developer Network document that discusses the process for viewing xMII content on mobile devices.

8 Security

Setting up Single Sign on between xMII & Enterprise Portal

9 Error Handling

Regardless of skill level, errors or exceptions are a fact of life for application developers. The disconnected nature of Web application development leaves many points where errors can and do occur. The key is gracefully handling any unexpected (or expected) errors to control the user experience. With JavaScript, there are various techniques available, as well as language features, to properly handle any problems. With xMII Transactions, there are several actions available to properly handle the errors occurred in the run of business logics.

Performing Error Handling with Web Page Generation (Especially on JavaScript):

DO/DON'T	Task	Why?
Generic JavaScript		
DO	Use Web page editor such as FrontPage, DreamWeaver to edit Web pages.	To avoid the potential typo or other errors at design time.
DO	Check everything before proceeding. That is, validate an object, method call and assignment of variable and so forth before utilizing them. Use alert to raise warnings or errors.	This avoids any errors associated with working with an object that has not been instantiated or a call to a method that does not exist or assignment value that is not valid.
DO	Use try/catch/finally statement in JavaScript. This statement encloses a block of code in which an exception, like a runtime error, may occur. The catch clause outlines how the error will be handled, and the finally block includes code that is always executed.	The try/catch/finally code tries to execute a block of code and control is passed to the catch block if it does not execute successfully. The catch block is skipped if no errors occur. The finally block executes after the try and catch blocks finish.
JavaScript with xMII		

DO	Use executeCommand method in JavaScript when interact with iCommand Applet, such as:
	`if (document.Applet.executeCommand()) {`
	`    alert("Successful");`
	`} else {`
	`    alert("Failed with error: " + document.Applet.getLastError());`
	`}`
		This method is used to execute the command associated with the iCommand applet's query template. It returns true on a success, and false if an error occurred.

DO	Use .getLastError() method in JavaScript when interact with iCommand Applet. (see prior example)	This method is used to return a string describing the last error associated with a call to the executeCommand() method.

DO/DON'T	Task	Why?

DO	Use the following query object methods to provide query status:
	isDataValid();
	getLastStatusCode();
	getStatusMessage();
		isDataValid(); is a validity flag that shows "true" or "false."
		getLastStatusCode(); shows a numeric status code.
		getStatusMessage(); shows the last message.

Performing Error Handling with xMII Transactions:

DO/DON'T	Task	Why?
DO	User Trace Action and trace log or other logs to debug at the development time.	The Tracer action is used to help in debugging. Trace logs provide step-by-step information. The general log is also a good place to find typical debugging information.
DO	Use Event Logger Action to record important events and errors within a transaction.	The Event Logger Action allows developer to create an entry in the User Log. User can define the event type and message. This can be a useful way to record important events and errors within a transaction.
DO	Use xMII Fatal Error Action if you want SAP xMII to handle the output of a transaction as a fatal error.	Fatal Message action is designed to create an SAP xMII document with a single failure message. On a non-recoverable, or fatal, error, SAP xMII assumes a single message will be returned.
DO	Use xMII Informational Message Action if you want SAP xMII to handle the output of a transaction as an informational message.	Informational Message action is designed to create add *one or more* informational messages to an SAP xMII document. This type of message is used to add additional information on an incomplete result set, or to add context to a data set, without the need to handle the result as a non-recoverable, or fatal, error.
DO	Use xMII Terminate Transaction Action to terminate a transaction at a specific point in the execution with Termination Message specified.	Terminate Transaction action is extremely helpful when testing specific sequences of actions, without having to execute the entire transaction. It can also be used in conjunction with a conditional action to terminate the execution of the transaction provided that a specific condition is met. When configured to Terminate With error, the Termination Message is used as the error message returned by the transaction.

DO/DON'T	Task	Why?
DO	Use LastErrorMessage Property to debug and error handle since it will contain the response message from the configured Transaction Call.	If a Terminate Transaction Action is encountered within the configured transaction, the configured termination message will be returned to the LastErrorMessage property.
DO	Handle exceptions in the response document received from the called function module when use SAP ERP Interface Actions such as JCO, WAS, BC interface actions.	The SAP ERP Interface actions are used to send XML messages to and from SAP. Depending on the type of functions (BAPI, RFC, etc.), return errors and specific error messages may be in different paths within the XML, particularly in custom built function modules. There are different types of message returns (errors, warnings, informational messages, and success), and each response can have one or more return messages. Exception handling should be processed based on these specific message returns.

10 Go Live! Check List

10.1 Final Application Readiness Check List

- ☐ Make sure the log settings (Log Management…Log Configuration) are set to either Warning or Error.
- ☐ If using any UDS type connections, make sure that any Debug settings have been turned off.
- ☐ Perform full application backup, including system and user configuration, all application content including query and display templates, web content, business logic services transactions, reference docs and schedules.
- ☐ Eliminate any unused or empty backup or temp folders in any of the pertinent content locations.
- ☐ Eliminate any backup files or debugging content files found in any of the pertinent content locations.
- ☐ Make sure that any transactions using unnecessary XMLSaver action blocks for debugging purposes are either disabled with a false conditional block or are removed from the TRX.
- ☐ Comment out or remove any unnecessary debug type JavaScript alert messages from the web pages.
- ☐ Correct any non-relative page links in the web content, especially for fully qualified URLs.
- ☐ Look for APPLET definitions that contain non-standard attributes, proper format for iChart is as follows: CODEBASE="/Illuminator/Classes" CODE="iChart" ARCHIVE="illum8.zip"
- ☐ Review Business Logic schedules; remove unnecessary ones and adjusting time interval configurations where applicable.

10.2 Final Application Click-Through

- ☐ Do not use http://localhost to reference the server for the click-through exercise, instead use an actual client imaged PC so as to emulate the user experience and expose any potential authentication problems.
- ☐ Do not log-in with the Admin user, but effectively test the authentication and authorization by using a configured user account.
- ☐ When using additional login accounts to test various levels of security and content make sure to close all open browsers to ensure a fresh login session.
- ☐ Monitor the Sun Java Console during the site click-through looking for pages that may have potential issues.
- ☐ Watch for errors in the browser's status bar, including potential script errors. This can be enhanced by using the Internet Explorer Advanced setting for browsing: 'Display a notification about every script error'.
- ☐ While doing the click-through either use the interactive Log Monitor application (Log Management section in the Menu), or periodically check the Logs using the appropriate viewer page.

11 References… More Help!

The following outline will document the many available resources to resolve problems, find relevant information and simply to find additional information regarding a certain SAP component or product.

SAP Service Marketplace (http://service.sap.com)

The focus of this site is to provide a source for many portals that can deliver information on specific content.

SAP Developer Network (http://www.sdn.sap.com)

"SAP Developer Network (SDN) is an active online community where ABAP, Java, .NET, and other cutting-edge technologies converge to form a resource and collaboration channel for SAP developers, consultants, integrators, and business analysts. SDN hosts a technical library, expert blogs, exclusive downloads and code samples, an extensive eLearning catalog, and active, moderated discussion forums." (https://www.sdn.sap.com/irj/sdn/about)

SAP Help Portal (http://help.sap.com)

A great source for web-based documentation that covers all SAP Solutions. You can search and browse for help guides dealing with virtually every component of SAP ranging from xApps to R/3.

xMII: http://help.sap.com/saphelp_xmii115/helpdata/en/index.htm

Interface Repository (IFR) (http://ifr.sap.com)

A collection of all published SAP interfaces which are in agreement with W3C standards. This source is great for viewing the structure of BAPIs, RFCs, IDOCs, etc. as well as the definitions of each attribute associated with them.

Copyright