SAP Manufacturing Integration and Intelligence 12.X Best Practice Guide

Applies to:

SAP Manufacturing Integration and Intelligence (MII) versions 12.0 and 12.1

Summary

This guide identifies approaches for the most flexible and best performing MII applications.

Author: SAP MII Field Enablement and Support Teams

Company: SAP

Created on: July 15, 2009

Table of Contents

Introduction

This document shows high level techniques that standardize use of the product and for future troubleshooting and organization. Due to the varying functionality and flexibility of design with SAP MII, it is beneficial to have a guide for structuring the XHTML/HTML/IRPT content, as well as editing JavaScript and CSS in a standard manner.

This Best Practice Guide presents information on MII 12.X, but some sections will provide more information on specific versions.

Architecture

SAP NetWeaver

MII 12.0 was released on the SAP NetWeaver platform with the Web Application Server (Web AS) J2EE architecture. The 12.0 release of MII eliminates the dependency on the Microsoft Windows operating system, Internet Information Services (IIS), and NewAtlanta's ServletExec Application Server.

SAP MII, on NetWeaver WebAS, provides the customer with scalability, high availability, load balancing, and built-in cluster support. The NetWeaver database backend component is leveraged by MII for all configuration and application component objects. WebAS data sources for JDBC connection pools are available in addition to the native MII IDBC Data Server connections to relational databases.

The NetWeaver User Management Engine (UME) replaces the MII LHSecurity application component used in prior versions. The UME security modules, LDAP, Databases, local UME users, SAP SSO2, Kerberos, and others, provide integration with a wide array of customer security models. For MII version 11.5 customers, the Migration Tool provides an option for migrating native MII LHSecurity users and roles into UME as members of the Database Data Source.

For MII 12.1, the SAP NetWeaver Composition Environment (SAP NetWeaver CE) allows developers to create custom applications. The open-source Eclipse Platform solution provides an integrated development environment for building all layers of a composite application. With a robust Java Platform and an integrated build-management development infrastructure, the SAP NetWeaver Application Server (SAP NetWeaver AS) provides a runtime environment for composite applications and supports the Java EE 5 specification.

Also in MII 12.1, developers model enterprise services and interfaces, and store their metadata in the central Enterprise Services Repository. The repository includes thousands of services provided by SAP as well as any services you add to it. These enterprise services adhere to a coherent and consistent set of semantics designed to support business process automation and are built on open standards, such as global data types. An integrated development infrastructure supports the complete life-cycle management of composite applications. It offers a component-based build and transport infrastructure. Optional source control for MII 12.1 delivers version control over MII content.

MII 12.0 uses NetWeaver Application Server 7.0 (2004s) Java SPS 14 or higher

MII 12.1 uses NetWeaver Composition Environment (CE) 7.1.1 EHP1 or higher

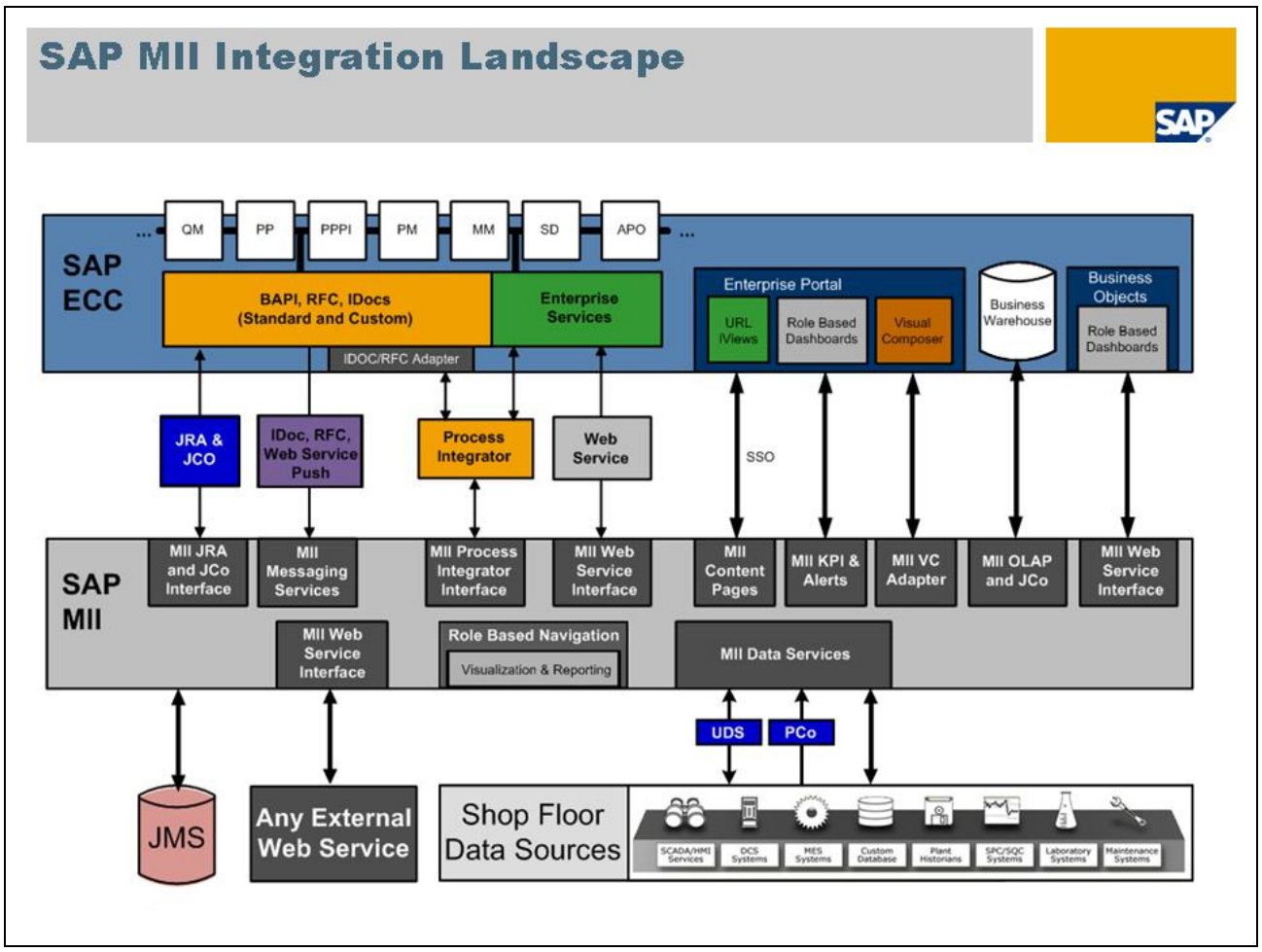
* For more details on MII versions, go to http://service.sap.com/pam

MII Integration Architecture

MII integrates with manufacturing systems and SAP ERP Central Component (ECC) or R/3 connecting the shop floor operation level and enterprise data. This allows you to monitor your operations and resolve manufacturing exceptions in real time.

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

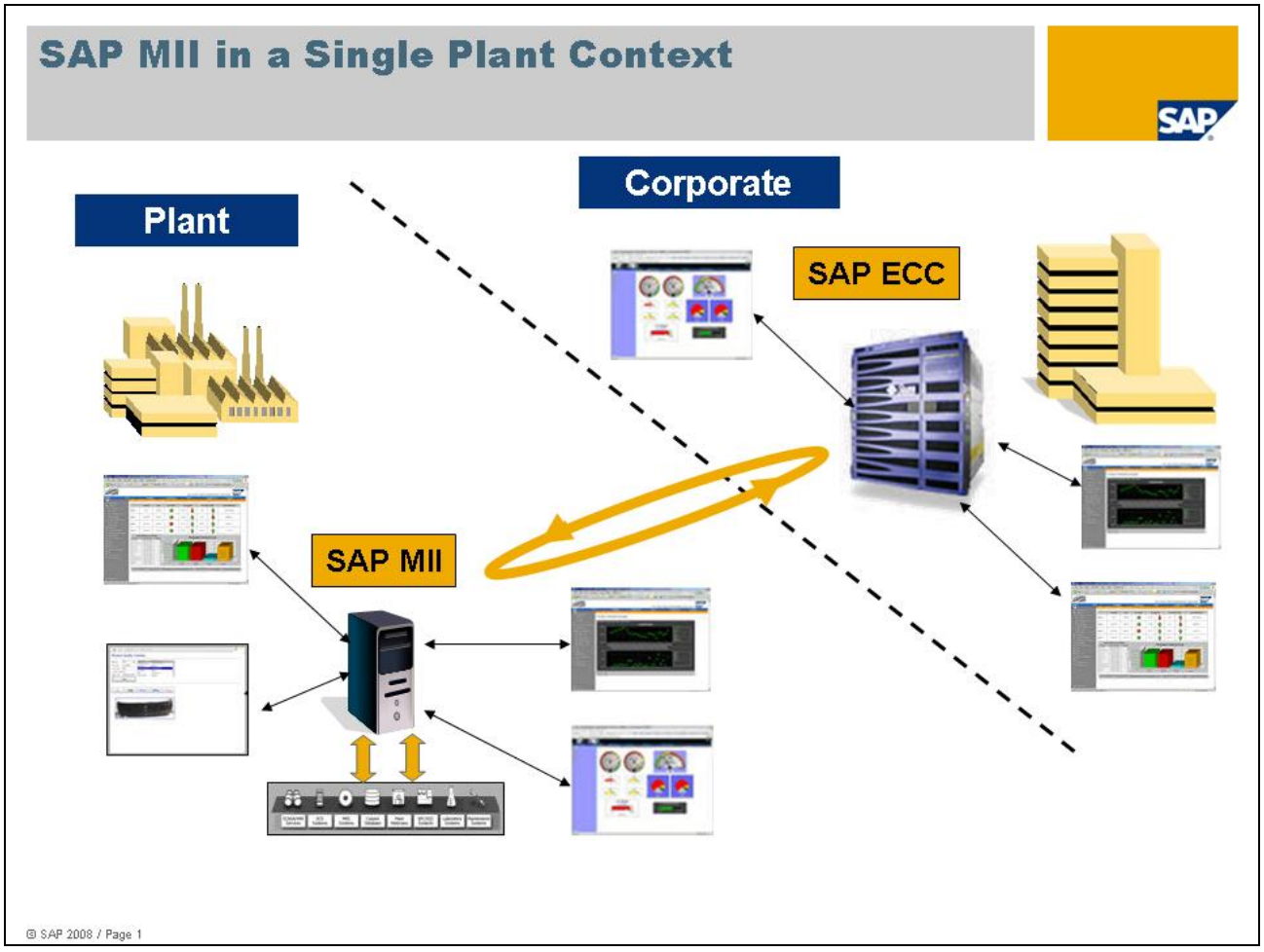
Access to 100% of ERP functionality integration is supported through 3rd Party Middleware/EAI products as well.



SAP MII Single Plant Architecture

In the single plant example below, an MII server is installed on the plant network to provide operational integration and visibility for production personnel. At most plants, MII connects to data sources, data historians, control systems, etc. that can generate very large data volumes. With MII located at the plant, there is minimal performance impact with respect to the wide area network (WAN). In the event of a WAN outage or ERP maintenance, MII can continue to operate while disconnected from ERP.

Connecting to plant data sources can be a complex combination of software and network choices. Firewalls, ports, routers, UDS, DNS, OLEDB, OPCDA/OPCHDA, O/S, hardware, and security are some common factors among the many considerations when making the connections. The subject is narrow in focus, but broad in complexity and for the purposes of this document, out of scope. Nonetheless, those connections will be necessary in your architecture.

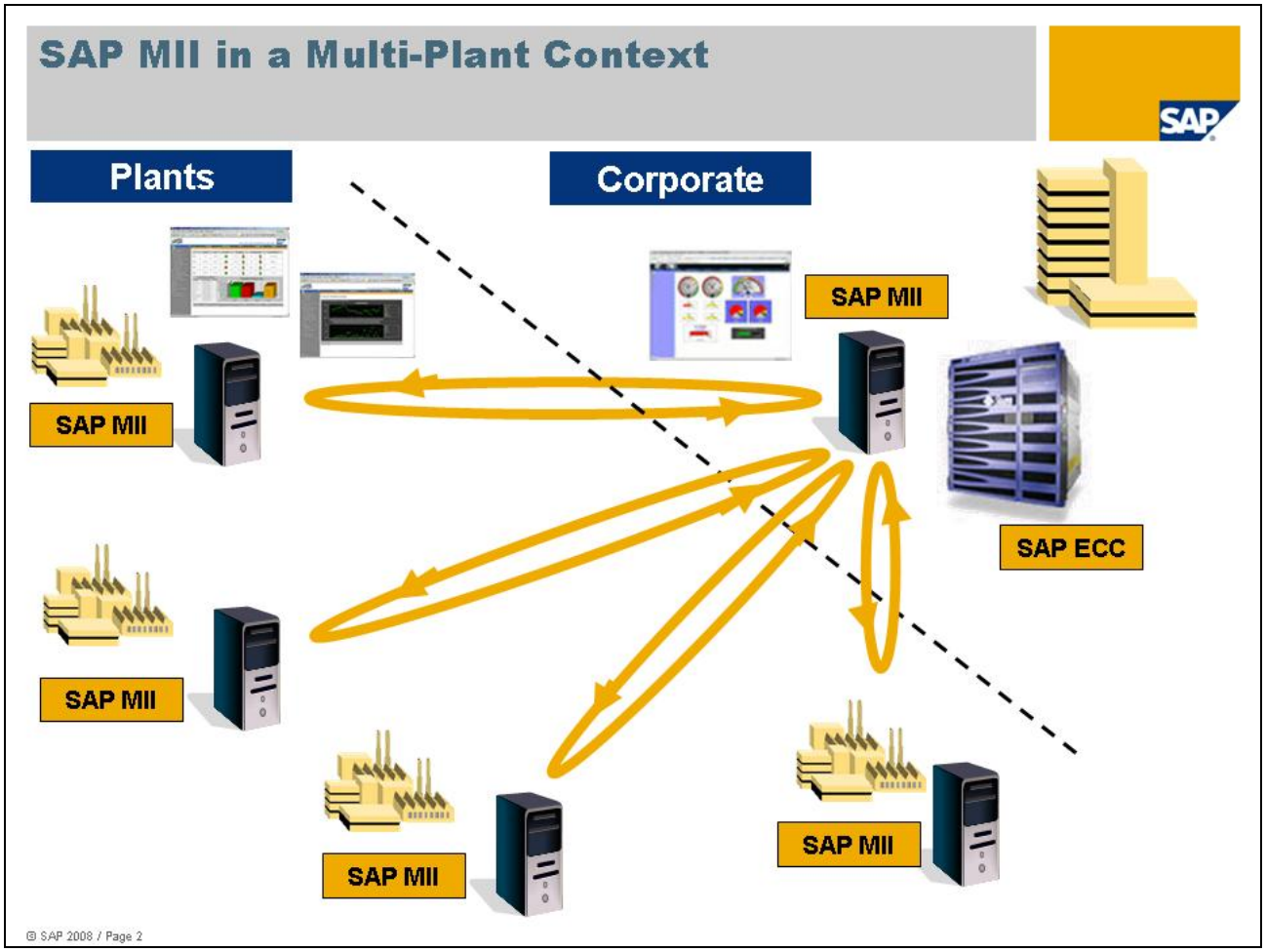


SAP MII Enterprise Architecture

In the example below, each plant site has an MII server. An additional MII corporate server processes multi-site data to provide plant-to-plant or divisional analytics for corporate scorecards, dashboards and reports. With a central instance, virtual server connections to the plant MII instances use binary data streams to extract data without local replication. Without a central server, the Multi-Plant context is simply multiple Single Plants.

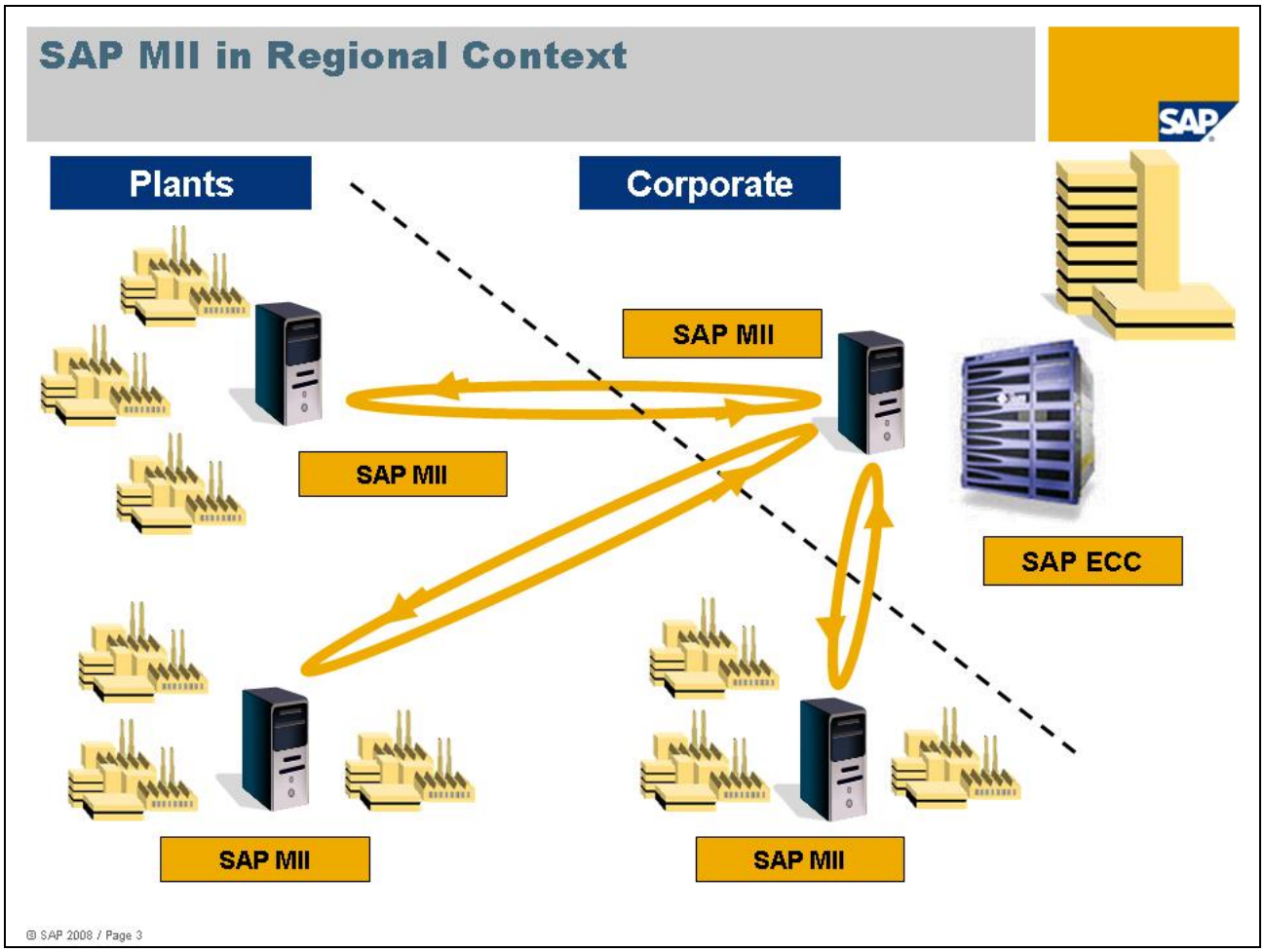
Whether, and how, to use a central, corporate MII server depends upon business requirements and the system landscape. One company's architecture may be different than another's despite similar applications. The two different architectures may both be "right" for their circumstances. However with a real or near-real time corporate level dashboard business requirement, a central MII server should be considered. If the dashboard requirements are not near-real time, BI reports may be more applicable.

ECC connectivity is generally recommended to be directly with the plant MII instances rather than through the central MII instance. However, the ECC traffic can be routed through the central MII instance. The main tradeoff here is single point failure of the central MII instance versus improved WAN performance. Sometimes a limited WAN capacity is the greater concern.



SAP MII Enterprise Architecture for Regional Landscapes

When there are clusters of plants with a common datacenter or several small plants in a small geographic area, a single MII instance can support more than one plant effectively. From an MII central instance, the comments above in the Multi-Plant context still apply.



SAP MII Single Central Landscapes (General DON'Ts)

Often for cost considerations, a single central MII server (without plant MII servers) is suggested as the architecture. Initially it can be used to demonstrate a Proof of Concept though performance can be substantially degraded by accessing large datasets across the WAN. Others have used this architecture for a simplified GUI based on ECC (or other central data sources). The main consideration is to have the MII server close to the main sources of data. Usually, that means at the plant near the manufacturing data (MES, data historians, etc.). Consider carefully the WAN loading, data time zone implications, scalability, disconnected operation capability, and response time when deciding upon your architecture.

Server Configuration Best Practices

Web Servers

Each MII site (plant or plant cluster) should have at least a Development and Production server. A third, QA, instance is desirable. The two (or three) server configuration and installations should be identical for fast and easy promotion from Development (to QA) to Production environments.

Web Applications

Definitions

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

Web Application Server – the back-end servlet engine used to supplement the functions of a web application; SAP NetWeaver

Deployment Strategies

Please view the recommended NetWeaver deployment practices located in **http://help.sap.com/NetWeaver** or through the **http://service.sap.com/instguides** link.

MII can be deployed on virtual server as well as physical server configurations because of NetWeaver's virtualization capabilities. There are no MII configuration implications between the NetWeaver and MII platforms. For more information on NetWeaver virtualization, visit this link.

Installation & Migration

The recommendations in this section require careful consideration. Understand the implications thoroughly before deciding to ignore these recommendations.

Recommendations

The installation guide in the SAP Support Portal (**http://service.sap.com/instguides**) should be reviewed and fully understood BEFORE anything is done. It will require you to sign in with your S number Id. Also, check if there are any release notes that apply to the MII version you are installing.

Each instance of SAP MII should have at least a development and a production instance. Perform a full backup of your development and production instances before any upgrade is performed.

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

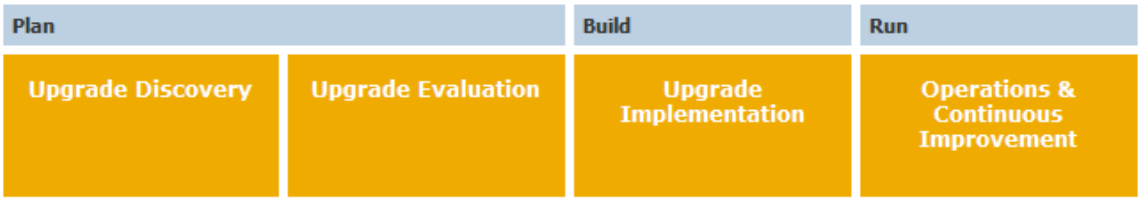
Information about all customization should be recorded. This includes all custom action blocks, customized style sheets, customized themes, custom SVA/SVG images, custom reference documents, and any other customization done.

If you are upgrading to MII 12.0, please be aware that new custom action blocks must be written in Java (MII 11.5 and newer require code in Java).

For more of an in-depth look at upgrading from MII 11.5 to MII 12.0 go to this **link**.

Performing the Upgrade

An upgrade should follow the Plan-Build-Run methodology found on http://service.sap.com/upgrade .



Follow the specific MII upgrade procedures in the Upgrade Master Guide on http://service.sap.com/instguides.

Begin with the development instance in your landscape. Once upgraded, the development server should be thoroughly tested before the upgrade is performed on the production instance. The recommended time for this testing period depends on the developed application, but at a minimum should include a complete click through of the entire application and its functionality.

If you have any issues with your SAP MII server instance during the installation, please consult the NetWeaver logs first before contacting support. If you have issues when migrating, please consult the migration log first before contacting support.

Once the upgraded development server has been 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.

If virtual server connections were used for server to server connectivity between corporate and plant instances, then 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 MII product can be made.

Application Transports

Organizations should follow the previously mentioned server configurations and landscape best practices. When transporting an application from a development to a test/QA environment, the administrator should export a project (see Project Best Practices) and save the zipped project to a safe location. After completing the backup, the administrator will simply import the project into the test/QA system. Projects and their folders work as an insert/update to the database. For example, if a project already exists the system will just update the project with what has changed. If the project is nonexistent, then it will do an insertion. Deletion only occurs manually. The procedure will then be repeated for the production environment. Applications should be clicked through and the NetWeaver logs should be checked to verify all MII objects are working correctly.

Source Control in MII 12.1

In MII 12.1, the Source Control Service can help you have a smoother development process by managing and coordinating multiple developers' work for a solution. This feature is focused on managed version control of MII content**.** Activities are used to group changes for versioned content. A developer can check in an Activity, check out Activities, view version history on an MII object, revert back to a previous version and synchronize different versions.

Source Control is set up first in NetWeaver Development Infrastructure (NWDI). For more information on NWDI, please visit http://help.sap.com/NetWeaver . An administrator must configure it in the Source Control Service category in MII.

In order to modify the NWDI User Configuration screen in MII, the user must have a SAP XMII Super Admin role and belong to the NWDI Developer Group.

The Source Control Service in MII is not a replacement for well designed development change management processes.

DO/DON'T	Task	Why?
DO	Create and check in an Activity for the sole purpose of creating of new shared project. Then, create one or more Activities to create necessary sub folders and MII objects.	Reverting an Activity based on an incorrect query change can cause the project name to be taken up in NWDI.
DO	Check out a web page's corresponding applet's query and display templates when modifying a web page.	One developer may change the query parameters while another overwrites the parameters in the web page causing rework.
DO	Use good naming conventions and descriptions for Activities.	Names should give insight to the purpose of the Activity. Descriptions could be associated with your business processes (support tickets, enhancement requests, project work packages, etc.).
DO	View open Activities and make sure all that are open are resolved before migrating your application to the next track in your organization (dev, test, prod).	NWDI will not allow you to migrate an application that contains open Activities.

Application Design Best Practices

Naming Convention Best Practices

Naming conventions are one of the most important aspects when developing MII applications. Naming conventions may differ from the ones below to due strict company standards or regulations. However, it is very important to be very consistent when using meaningful naming conventions.

MII Objects

All templates should be named in the following manner:

```
<SpecificPurpose><ObjectType>
```

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. Developers should use the Camel Case notation for MII objects in this example (first letter of a word is capitalized in an identifier). Other examples are:

```
ShiftDetailTransaction
ShiftDetailChart
ShiftDetailGrid
```

Keep in mind that template names are always provided as fully qualified paths, so the directory structure, which includes the project name and subfolders, carries additional information regarding the template as in this example:

```
CompanyABC/Metrics/OEE/ShiftDetailQuery
```

Good naming conventions will make it easier for developers to find and check out corresponding objects when using MII 12.1 Content Management features. See the Application Transport section.

JavaScript and Web Related

Consider the creation of JavaScript variables to reference the Applet object, Display Object, and Query Object as follows (using the Pascal case naming convention where first word is not capitalized but other words are capitalized):

```
var myApplet = document.iGrid;
var myGrid = myApplet.getGridObject();
var myQuery = myApplet.getQueryObject();
```

Naming an applet in an IRPT or HTML page that corresponds to its object is another good practice. For example:

```
<APPLET NAME="AssetUtilGrid" …>
```

This describes the applet's purpose when troubleshooting a page. At a quick glance, a developer can see that the iGrid applet displays values for asset utilization. Also, if the applet has an issue, a developer can look at the Java Console and see what errors are thrown for a particular applet.

Projects

Projects should be either named after the company or the company division (which depends on how many MII applications are present). Folders and sub folders should be named for applications and functional areas.

Example:

```
CompanyDivision\AppName1\FunctionalSet1
```

In MII 12.1, Content Management allows for the versioning of project content via a check in and check out of Activities. Naming of Activities should be defined by the organization's business processes. Activity names could be derived from enhancement requests, problem tickets, or project work packages to name a few examples.

Data Servers

Data server naming in MII should be kept simple and general. Where possible, avoid using site specific or development level names. The name of the actual data server is stored within a saved query template, whereas the data server description is not, so keep the names simple and the descriptions detailed, especially in a complex environment with lots of data servers.

Project Best Practices

Project Definition

A Project is a root level container for MII queries, displays, transactions, web content, and related objects. Organizing content in a project structure has important administrative implications:

- Projects can be imported and exported
- Properties can be shared among transactions within a project (version 12.1)
- Content Management is administered at the Project level (version 12.1)

When a Project is created, it is present across the tabbed areas: Catalog and Web (and Meta-Inf, introduced in version 12.1). At the root level is the Project. A project contains folders (which can themselves contain folders). The Catalog tab shows MII query templates, display templates, transactions, reference documents, animated objects, etc. The Web tab shows all web-related content such as HTML, IRPT, XML, CSS, images, etc. The Meta-Inf Tab is new in Version 12.1, and this tab shows Shared Properties and Localization tokens.

Maintaining a mirrored folder structure for the Catalog, Web, and Meta-Inf tabs will provide overall organization and logical traceability for all functionally dependant project component objects.

Commonality in Project Organization

It is recommended that a Common Project be created for objects shared across Projects (if there are multiple company projects and some basic MII commonality among them). Examples of common objects are as follows:

- Utility queries and transactions
- Common graphics and other images
- Company themed display templates
- Reusable JavaScript library files
- Cascading Style Sheets (CSS)

A Common folder could also be created in the company project or application folder and used to store commonly used objects that are customized for a project. Setting up a common folder at this level rather than creating a common project depends on the size of your MII applications and how much can be consolidated for reuse.

Example:

```
CompanyDivision\Common\iGrid
CompanyDivision\Common\iCalendar
```

An MII lead architect should be responsible for the content of common directories. There should be a submittal and approval process before common objects are used by all application developers.

In MII 12.1, the Content Management functionality provides change management features which track modifications to files including items in the common folder in the project.

Project DOs and DON'Ts

DO/DON'T	Task	Why?
DO	Create a Common Project for objects shared across Projects. A Common folder can be created within a Project itself for content shared at the project level.	Leveraging a Common Project or folder promotes reuse, consistency, eliminates duplication, and saves development time (and disk space!).
DO	Mirror the folder structure on the Catalog, Web, and Meta-Inf tabs to the greatest extent possible.	Consistency in structure helps developers locate needed MII objects.
DO	Use good naming conventions when naming projects and folders.	See Naming Conventions for more information.
DO	Load the appropriate template, and use the Save As function to save the template to the desired application directory, where content specific changes can then be made.	Templates in a common folder will insure common colors, font style and sizes, and properties for all newly developed content.
DO	Create reusable content that allows for dynamic query and transactional capabilities of MII.	Promotes more flexible and less redundant content.
DO	Create a folder structure in a project for each functional area of an MII web application. Example: `CompanyDivision\AppName1\FunctionalSet1`	Provides organization and structure to the application itself.
DON'T	Create a separate folder for queries, a separate folder for charts, a separate folder for grids, etc., but rather group related objects within the same folder.	It is much easier to find the components of an application when stored as a functional area rather than by object type.
DON'T	Leave old unnecessary backups of MII objects and folders in your projects.	Extra unused content and folders can lead to confusing, incorrect links in display templates and web pages.

Structure DOs & DON'Ts

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

DO/DON'T	Task	Why?
DO	Use consistent JavaScript naming conventions.	See Naming Convention section.
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	Leverage applet events properly. Drive multiple applet loading from MII 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. See **SAP Note 1314315**.
DO	Evaluate each MII Applet definition; Take advantage of the supplied MII 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.
DO	Follow a modularized design approach and load only applets that are needed. Using iFrames and `<PARAM NAME="InitialUpdate" VALUE="false">` also follows this approach.	Loading all applets at once increases user wait time for web pages and can result in performance issues depending on data set size. Hiding and showing applets can also impact performance. Successful use of applet events allows users to drill down to see more information on particular details.
DON'T	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.

Presentation DOs & DON'Ts

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, all web pages should contain references to common Cascading Style Sheets.

A common cascading stylesheet library should be used to keep the same look and feel between web pages within the MII web application. For a project starting point, the customized stylesheet can be copied to the Common project directory underneath the company project 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. There are exceptions to this rule, but using external style sheet is generally recommended.

Behavior DOs & DON'Ts

Behavior 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.

DO/DON'T	Task	Why?
DO	Develop data driven pages (ex. IRPT pages) in an application rather than HTML pages when necessary.	Dynamic web pages improve application flexibility and robustness. Less hard coded functionality also reduces application maintenance.
DO	All variables need to be declared; it is good practice to declare them at the top of a function. 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. Try to use comments to explain the "why" and not the "how".	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 MII Script Assistant to generate and determine correct query/display object methods and properties.	The MII 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. In MII 12.1, the Workbench contains built-in script and applet assistants which are recommended for use.
DO	Utilize Session Properties for static properties.	See the Session Variables section.
DO	Utilize URL properties when passing values from page to page.	URL properties are more efficient that Session properties in this scenario. See the Session Variables section.
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.	It is recommended 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.

With JavaScript, there are various techniques available, as well as language features, to properly handle any problems. Performing Error Handling with Web Page Generation (especially with JavaScript):

DO/DON'T	Task	Why?
Generic JavaScript		
DO	Use the MII Workbench or a web page editor 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	Handle all exceptions by using the 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 MII		

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	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.

DO	Use the appropriate method to update objects with an underlying time based query.	In the case of a time-based query, the refresh() method will update the start and end dates based on the current time less the duration/duration units, but the updateChart (iGrid, iSPCChart, etc.) method will use the specified dates.

Transaction DOs & DON'Ts

DO/DON'T	Task	Why?
DO	Create modular transactions that are as simplistic as possible and yet meet requirements. (* See DON'T section for exceptions to this regarding XML documents)	Smaller and simpler transactions are easier to debug and unit test. Developers should consider the size and impact of their transactions.
DO	Use default values for debugging transactions.	Setting default values allow a developer to initially test transactions and remove issues before issues are found in run time.
DO	Leverage Reference Documents.	Helps avoid linking mistakes and aids in building transactions.
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 file I/O that is 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	Use the reference arrows on action blocks for a visual indicator of incoming and outgoing parameter assignments.	Easy method of quickly determining an action's configuration.
DO	Leverage XPath functionality when looping through a dataset within a transaction.	XPath can dramatically increase performance gains for looping conditions.
DO	Use the MII protocol identifiers (server://, db://, web://) rather than using fully qualified paths when accessing and publishing files.	Transportation issues can impact the application when C:/… is not the same in your different MII landscapes.
DO	Declare date parameters in the native DateTime format (not as a string).	Will make date-time calculations much easier and faster.

DO/DON'T	Task	Why?
DON'T	Pass large XML segments from transaction to transaction and from action block to action block.	A copy of the XML gets created for each handoff. Large XML documents can negatively affect performance. In 12.1, use Shared Properties to avoid duplicating large XML documents.
DON'T	Use large XML documents as default values for ease in linking and debugging.	The large XML documents will be saved as part of the transaction and can adversely affect load time and performance. Use small reference versions for debugging or clear defaults after debugging is complete.
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.

With MII Transactions, there are several actions available to properly handle the errors occurred in the run of business logics. Performing Error Handling with MII 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 MII Fatal Error Action if you want SAP MII to handle the output of a transaction as a fatal error.	Fatal Message action is designed to create an SAP MII document with a single failure message. On a non-recoverable, or fatal, error, SAP MII assumes a single message will be returned.
DO	Use SAP MII XML Output Message Action if you want SAP MII to handle the output of a transaction as an informational message.	The Message action is designed to create add *one or more* informational messages to an SAP MII 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 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 provides the error message returned by the transaction.
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/DON'T	Task	Why?
DO	Handle exceptions in the response document received from the called function module when use SAP ERP Interface Actions such as JRA, 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.

Tips and Tricks for Building ERP Interfaces in MII – document providing information on using interfaces between ECC and MII transactions

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

Other MII Best Practices

The following points are general MII 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 MII for maximum usability and response time.

Querying & Caching

Properly used query caching in the MII environment greatly improves the efficiency of your MII application.

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 at which the user requires the data to be viewed. 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.

Use the MII cache where the data values change once a day or even once a week. In this situation, the data coming back from the data source (e.g. data warehouse) may take longer than the user cares to wait for their reports. The large data set result leads to performance complaints and people becoming discouraged from using the interface. The solution would be to set the cache duration for a longer period of time to increase performance; however, there can be a data problem with setting the cache for long time durations. The query cache on your query may be set for twenty hours to accommodate people in different time zones. Depending on the user's time zone, they might still get information from the previous day. The link below will remove all of the cached values ensuring that there aren't any dataset values from the previous day.

To clear the MII system cache via a URL servlet call to the MII server use:
`/XMII/Illuminator?Service=QueryCaching&Mode=ClearCache`

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. For example, 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 MII 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 MII 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 they are 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 MII 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.
DO	Create custom views in the database instead of writing complex queries in MII.	Data manipulation at or near the data source can be faster than through MII query capabilities. Databases can perform joins and filters faster.
DO	Design query templates with flexibility in mind by using `Param.x` and start dates & end dates (SD/ED).	Queries can be robust and reusable when parameters are utilized.
DON'T	Use date ranges (time sensitive) 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 MII cache.

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`, `IllumLoginName`, `IllumLoginRoles`, and `Language`. Other session variables can be added by using the Custom Attributes option under the System Management category in MII.

They can be viewed with `http://<servername>:<port>/XMII/PropertyAccessServlet?Mode=List`

Even though HTTP Request variables are not technically session variables (they are only usable by that page because they are not in the actual session), request variables can also be passed into .IRPT pages via the URL. The main advantage is that it makes fewer trips to the MII server and increases response time.

Example:

`http://<servername>:<port>/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 Custom Attributes screen to create new session variables and Custom Attribute Mapping screen to assign roles 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.
DO	Use URL properties when passing values from page to page.	It is more efficient to pass values this way than using Session properties.

Log Management

Log management is now done using the NetWeaver logs. All log files are stored in the NetWeaver database. Access the log files using the NetWeaver Web Interface.

To use the web interface, simply type in the URL for your environment: (`http://<server>:<port>/nwa`). With correct permissions, this will take you to the SAP NetWeaver Administrator page and from there you will find a link for logs and traces. For viewing MII logs, a user must have either NetWeaver read-only log roles of "`SAP_JAVA_NWADMIN_LOGVIEWER_ONLY`" or "`SAP_JAVA_NWADMIN_CENTRAL_READONLY`".

For more information on NetWeaver log functionality, you can go to **http://help.sap.com/NetWeaver** .

DO/DON'T	Task	Why?
DO	Filter logs by MII content and severity greater than or equal to Warning.	These settings allow an administrator to quickly see on MII logs and traces.
DO	Set severity top level log/tracer level to Error for both Categories and Tracing when configuring the MII logs in NWA.	This configuration rule will help administrators consistently be able to view severity levels in an organized way.

Applet Debugging

It is recommended to use applet debugging techniques to resolve any user interface issues that arise. One general technique is to right click on the applet and view the query template and display template settings. The display template settings can be modified and updated in real time, however they will not be saved once the session is ended. Also by right clicking the applet, you can view the underlying data as HTML to view the values that populate the applet. Viewing the data as XML allows you to see the "Rowsets/Rowset/Row" format that is used by MII. To do this simply right click the applet > Data > View Details + CTRL button. In 12.1, you can use the right click steps and then select the Export Raw Data as XML option.

Another technique to troubleshoot applet problems is to use the Java Console to show the actions that each applet makes. Using the following line in an HTML/IRPT page, allows a developer to trace applet queries and data posting (query time in milliseconds).

```
<PARAM NAME="Trace" VALUE="true">
```

To receive more information in the Java Console, a developer can use the Debug parameter to see how long a query loads, query row count, date formats, and many more pieces of applet information.

```
<PARAM NAME="Debug" VALUE="true">
```

Using good naming conventions for your web pages' applets will make it easier to find the corresponding information in the Java Console when Trace/Debug parameters are turned on.

In MII 12.1, the applet debugging option is built into the applet toolbar in an HTML/IRPT page. When the debug parameter is on, the applet will then write debug messages to the browser's Java Console. A simple refresh of the page will allow the console to be populated with the debug messages.

Go Live! Check List

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, 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="/XMII/Classes" CODE="iChart" ARCHIVE="illum8.zip"`
- ☐ Review Business Logic schedules; remove unnecessary ones and adjusting time interval configurations where applicable.

Final Application Click-Through

- ☐ For the click-through exercise, 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 one or more configured user accounts.
- ☐ 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 periodically check the NetWeaver logs using NetWeaver Administrator.

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. You can find SAP Notes, which contain information on different releases of software. Login authentication is required.

http://service.sap.com/instguides to download installation and technical guides

http://service.sap.com/pam to view available platforms for different MII versions

http://service.sap.com/notes to view notes applicable MII known issues and solutions

http://service.sap.com/quicksizer to size your NetWeaver instance

http://service.sap.com/message to put in problem tickets

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.

Link for MII Forum to research technical solutions and post questions in a community environment

Link for MII Wiki to download guides, tips and tricks, etc.

Link for MII Articles on various technical topics

Link for MII sample projects and tools

SAP Education (http://www.sap.com/education)

SAP Education provides a catalog of all training courses available including MII courses. Courses available are instructor lead, online learning for Ramp Up projects (RKT), and Online Knowledge Product (OKP).

http://service.sap.com/rkt

http://service.sap.com/okp

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

This website houses and makes available all online documentation (SAP Library) for SAP solutions. It also has additional information about documentation, education services, and information design at SAP.

Copyright