SAP Manufacturing Implementation Architecture

Applies to:

SAP Manufacturing Software. For more information, visit the Manufacturing homepage.

Summary

The purpose of this document is to explain the standard implementation practices for Manufacturing Integration & Intelligence, Manufacturing Execution, and Plant Connectivity. The specific focus is around why and how to use the various products provided by SAP and the technical & business features of how they co-exist with each other in your landscape for maximum ROI.

Author: Salvatore Castro

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

Company: SAP Labs, LLC

Created on: April 28, 2011

Author Bio

Salvatore Castro of SAP Labs has a Bachelors Degree in Computer Engineering and a Masters Degree in Computer Science both through the Rochester Institute of Technology. He is a member of the LoB Manufacturing Group under Frank Schuler.

Table of Contents

Overview

When working with SAP manufacturing software there can be lots of initial questions about how best to utilize and configure your software and the expected return on your investment. Without knowing the answers to these questions proper implementation and usage of these products can be difficult. The contents of this document help to answer these questions from the perspective of an enterprise systems architect in order to help identify the best possible implementation strategy for your environment.

The first and foremost question that needs to be answered is what is the ultimate goal of the implementation and how can I manage this project to meet that goal. This seems like a pretty straightforward idea but in reality this can be much more difficult than initially anticipated. This is primarily due to continually changing business requirements leading to "scope creep" of a project. This is the number one hindrance for any IT project and will lead to the majority issues and delays. As a result a clearly defined set of goals and scoping guidelines are paramount for a successful project. It is also important to keep in mind that you shouldn't be out to solve all problems in your first release only most important ones and then build your solution from there.

This document describes the relationship between the SAP Manufacturing Software Ecosystem from the point of view of the SAP MII product which is the backbone for manufacturing integration. It covers the MII implementation, rollout and positioning strategy going forward along with the overall ideas and concepts behind the product design and implementation paradigm.

Requirements Gathering

There are many reasons why technical systems need to be integrated together with the primary one being to improve reports given to the decision makers in order to make quick and informed decisions. Another need is the ability of blue-collar employees to quickly and efficiently execute their job function. Both of these types of scenarios are relevant for efficient Manufacturing but target two very different audiences and should be addressed as such.

Centralized v. Decentralized

When defining project requirements the types of systems involved and where they reside in your corporate landscape needs to first be outlined. Since each company is different and even locations within the same company are different there's no easy answer to this, and only options that need to be considered. The primary question for any implementation is, where do I install my Manufacturing software so that it requires the lowest TCO for my IT department while still meeting the necessary requirements of my business? There are two things to consider when answering this question a central or a distributed architecture. Since both are technically possible it's a matter of which one fits your needs the best. Additionally, the SAP recommended approach is to configure a distributed environment and this is for multiple reasons.

Consider the following example to help make your decision and to understand why the SAP recommendation is to implement this product in a distributed architecture. An employee at a facility in Texas, where there are shop floor system (data historian and a local database). At the corporate office, based in Europe, is the primary data center where the SAP ERP, PI, BW, and Business Objects systems are located. If the Manufacturing software is located in Europe then the corporate IT datacenter resources can physically maintain and host the system. However, for the plant employees to get a report or execute their job, they have to access a web server in Europe which will communicate with the shop floor systems in Texas. Then this data is sent back to the system in Europe, and then from there sent to the browser in Texas.

This as you can imagine poses a serious performance concern for the manufacturing employee to efficiently execute his job and creates unnecessary network load. If the Manufacturing software is installed locally in Texas then corporate IT will have to rely on local IT resources to maintain the system hardware but still has the ability to software the software and content remotely. In this distributed scenario the employees interact with a local web server that communicates with local systems and can interact with the system data in a more efficient manner. The scenario is outlined visually for you here:

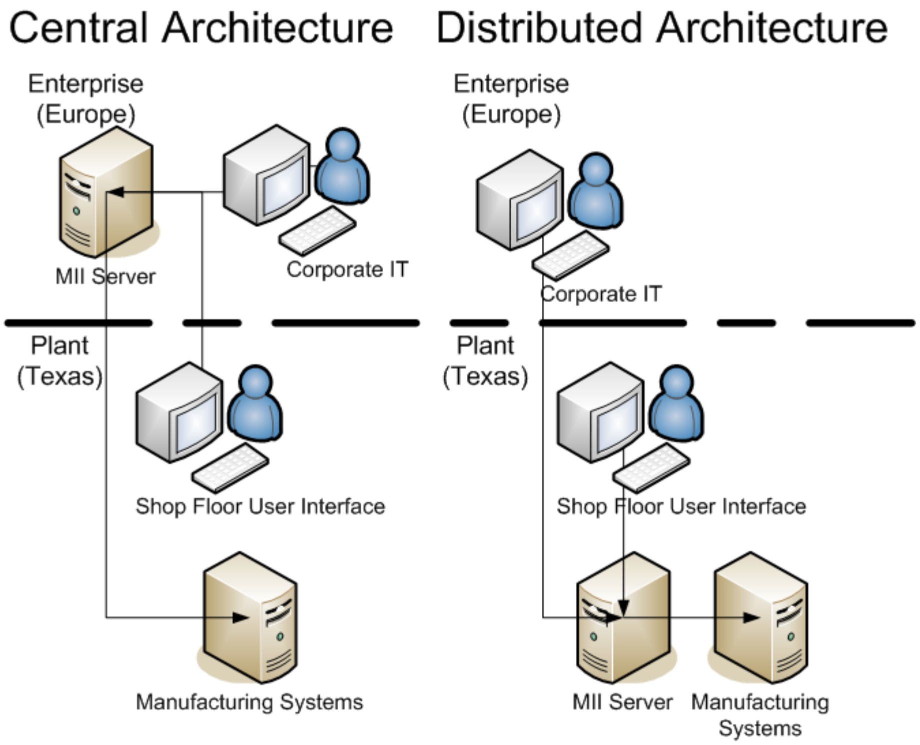

Figure 1: Network traffic in a centralized and distributed configuration

Content Creation

Deciding to build a web page that utilizes MII can be a simple or complex decision depending on the systems involved, experience with the MII solution, and complexity of the application. This is in addition to working with the various internal groups of your organization for continued support and maintenance of the application after it has been rolled-out.

Execution

When creating UI components it is necessary to define a clear set of UI requirements and also to explicitly define the boundaries between Execution & Reporting pages. The boundaries for this often span job function and role and need to be outlined first in order to deliver the best possible content for each job role. Simply setting a goal of “ERP Integration” will not yield the desired results of what the business needs even if it meets all of your initial project requirements. Often the existing Manufacturing Execution System (MES) will already have the UI content necessary for execution and the SAP Manufacturing Execution (ME) solution is no different. These existing web pages can be integrated into the Role based navigation environment provided by MII, this provides a single system login for the user base and helps with solution adoption and reduces your implementation overhead. Another reason for this is that there are no business rules defined in the MII product like there is in the SAP ERP, ME or a third party MES solution. This is because the MII product is designed to leverage your existing infrastructure and use the systems around it for business rules and data storage. If there is a need for execution pages to be built within the MII product it is to leverage the integration capabilities of the product for communication with these systems so that are synchronized with each other. The idea is to leverage the existing content where you can and define requirements around the gaps rather than replace what you have.

Execution requirements differ from reporting ones it that they often include large buttons for a touch screen, Statistical Process Control (SPC) Charts, near real-time information displayed on a “Job Board”, writes to and from plant and enterprise systems and finally a development technology that matches the available skills of your internal resources. This is usually a hot topic within a business because there are so many choices and groups that can be involved that there may not be a clear technology solution. This is where the flexibility of the MII web interface fits since it can utilize the vast majority of the currently available web technologies while generically interfacing to existing shop floor systems. In using this approach you can create a “Mash-Up” of your existing solutions that are enhanced by new development and combine them into a single common interface for the user.

Reporting

Reporting requirements may involve a wide range of requirements supporting a target audience that spans from high-level corporate to manufacturing operations reports. In order to facilitate this wide-ranging audience it’s important to identify what kind of data granularity is required and start there.

Enterprise & Local Reporting

Often it is the case that high-level corporate reports do not require any data directly from the manufacturing systems. This is typically the case where the MII product is rolled-out, because relevant manufacturing data is already synchronized with the enterprise ERP and BW systems. From there, tools and products from the Business Objects suite can be used for high-level KPI reporting. These high-level reports can link to local manufacturing reports hosted by an MII instance if the user chooses to drill-down into the more granular and near-real time operations data. This type of high level reporting with drill down capabilities is the most common reporting requirement and is achievable with a little bit of planning. The important part of this is to make sure that the data visible at the enterprise accurately represents what is going on at the lower levels. Subsequently the integration of the enterprise and manufacturing systems helps to ensure that this is the case.

Assembling Your Team

Like many companies out there, there are many enterprise reporting technologies that are currently being used by your business already. Plan to leverage these resources in your project for your high-level reports rather than trying to assemble a completely brand new team. From their skills, experience, and contacts you can find out why they are not happy with their current approach and obtain feedback about current issues to ensure the same mistakes are not made. This is a very generic concept but it will allow you to first engage the proper technical resources and get them interested and involved with the project early on in the process.

Reporting Technologies

For the local execution and reporting UI there are also a wide range of choices here but we have found that the simplest and easiest approach is to use the standard HTML and SAP MII Applets on these pages. There is a bit of a bias here but the applets do allow for a lot of flexibility in your application because it leverages standard HTML and JavaScript capabilities with the simplicity of the SAP MII Applets to do the data rendering. If this approach doesn’t work well for you then consider alternatives but make sure that they meet your reporting needs. There’s nothing that prohibits the use of multiple web technologies in your solution, with the exception of available implementation resources, and each technology brings something different to the end user experience. As an example, most web technologies do not support SPC Charting so a hybrid of SAP MII SPC charts and other web technologies can be used together (ie: HTML, Web Dynpro, Xcelsius, MS SilverLight, Adobe Flex, etc) in order to achieve a best of breed approach. Since the SAP MII application allows a REST based interface (WEB 2.0) to all of its content the use of other technologies can be achieved.

System Governance

When working in a distributed software implementation environment the number of systems to keep track of is significantly larger than in a single centralized implementation. SAP provides various tools and products to simplify this task so that software and content are both managed from a central location.

Maintain your SAP manufacturing software at each remote location is achieved through the use of Solution Manager, NWDI, and CTS+. The SAP Solution Manager product helps with managing the versions of software configured on the remote systems. The SAP NWDI and SAP CTS+ products help to manage the version of developed content running on the software and deployment of this content from Development to Consolidation (Optional) to QA and then to the Production system. This scenario is typical for customers performing central manufacturing content development and rolling this out to the local sites for productive use.

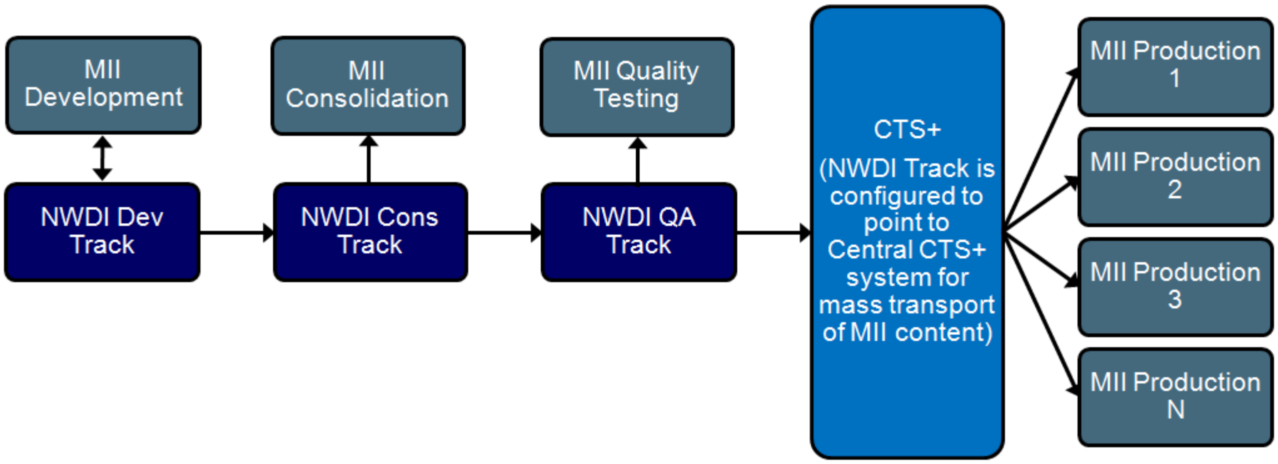

Figure 2: Transport and Management of Software using NWDI and CTS+

To expand on this scenario for more mature implementations, where manufacturing content development is done both centrally and locally, there is a replication of local NWDI tracks where central content is push to a local development server. Then both centrally and locally development content is promoted through local plant development, consolidation (Optional), QA, and production tracks.

Additionally, the health of the remote systems can be monitored by a central system through the use of the SAP CCMS tool. The SAP MII product is designed to seamlessly integrate with and provides content usage and performance information to this system for observation by the central IT staff.

There are various other tasks that are covered by the SAP manufacturing products, since they are based on the standard NetWeaver platform, and this includes user management, system configuration, and log viewing. The system parameter and configuration settings for the NetWeaver instance can also be configured through the browser based NetWeaver Administrator interface that is standard with all NetWeaver CE installations. The user administration for the SAP manufacturing product offering is based on the NetWeaver UME and accessible via the standard NetWeaver web front end for user management. Finally, the standard NetWeaver logs are where the applications write information about their execution providing a common view for all applications.

Given the system governance capabilities of the SAP manufacturing solution, the benefits of having local software systems typically outweigh any other concerns and costing overhead that may be generated by the additional hardware or IT resources required.

Typical Landscapes

A typical corporate landscape will include many different SAP Software systems, each with their own purpose. The SAP Solution Manager software must be installed in your corporate landscape in order to register your systems, download updates, and generate installation keys amongst a variety of other tasks. This product helps to reduce your overall software TCO and simplify administration of your SAP Software by allowing this task to be performed centrally.

Typical Architectures

The position of the various SAP software pieces relative to each other can be somewhat confusing and if not properly done can lead to maintenance and configuration issues. Here's a diagram of a typical implementation landscape including the following software pieces (Solution Manager, BI, PI, ECC on Java & ABAP stacks, LDAP, MII, ME, PCo, and existing Shop Floor Execution & Control systems). The non-SAP software components each operate in their own "silos" and can require multiple manual steps or costly development overhead to synchronize. This is where the SAP Process Integrator (PI) and SAP Manufacturing Integration & Intelligence (MII) software components fit into your landscape, each playing a different role in synchronizing these systems with a supported, consistent and clear approach. The PI system resides at the Enterprise level and is used to deliver reliable business to business messaging and large data transfer capabilities. For example when data is updated from the ECC to BI system for long term storage or when intermediate documents (IDocs) are generated which need to be delivered to external systems. The MII product typically exists at the Regional or Plant level and even at the Enterprise level to provide a distributed solution for reporting and synchronizing near real-time manufacturing information across your company. MII is designed to rapidly solve the inherent disconnect between your plant floor and enterprise systems and enabling your business to operate with a "Single Version of the Truth".

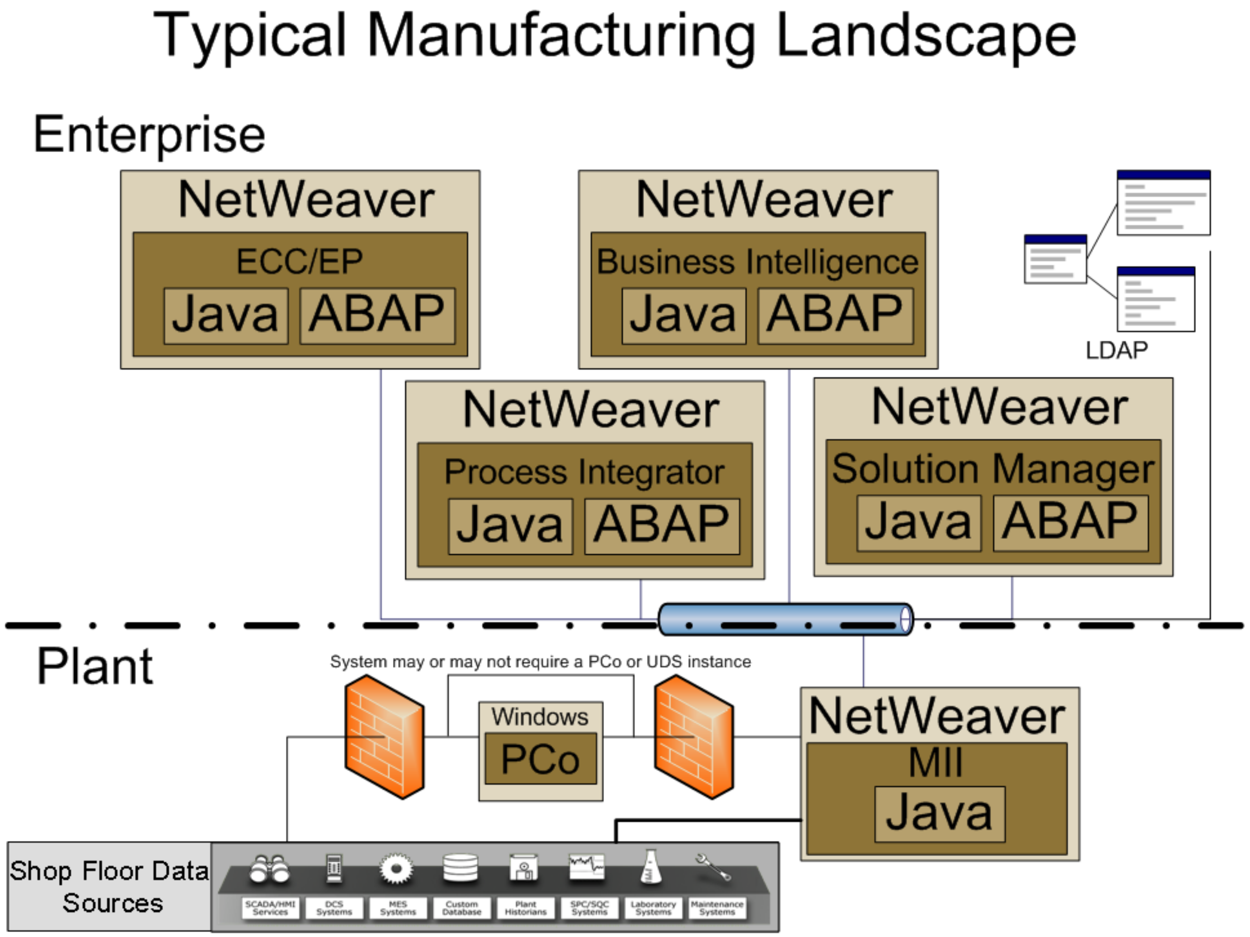

Figure 1-Typical SAP Software implementation landscape, the LDAP and Shop Floor Data Sources are 3rd Party Software Components. There may be other SAP Manufacturing products installed at the plant.

In the above diagram the SAP Manufacturing Execution (ME) software is considered one of many shop-floor data sources as the SAP MII product will work with any database centric system. With the SAP ME software product there are lots of benefits and out of the box features that come with this solution in addition to a well-defined and supported solution.

To go into further detail on what can be found at the manufacturing level, since there are a variety of systems from different vendors that are involved; the diagram below helps to highlight the various software levels. It outlines what a typical manufacturing environment looks like along with starting to highlight the level of system disconnects that are created when dealing with a manufacturing environment.

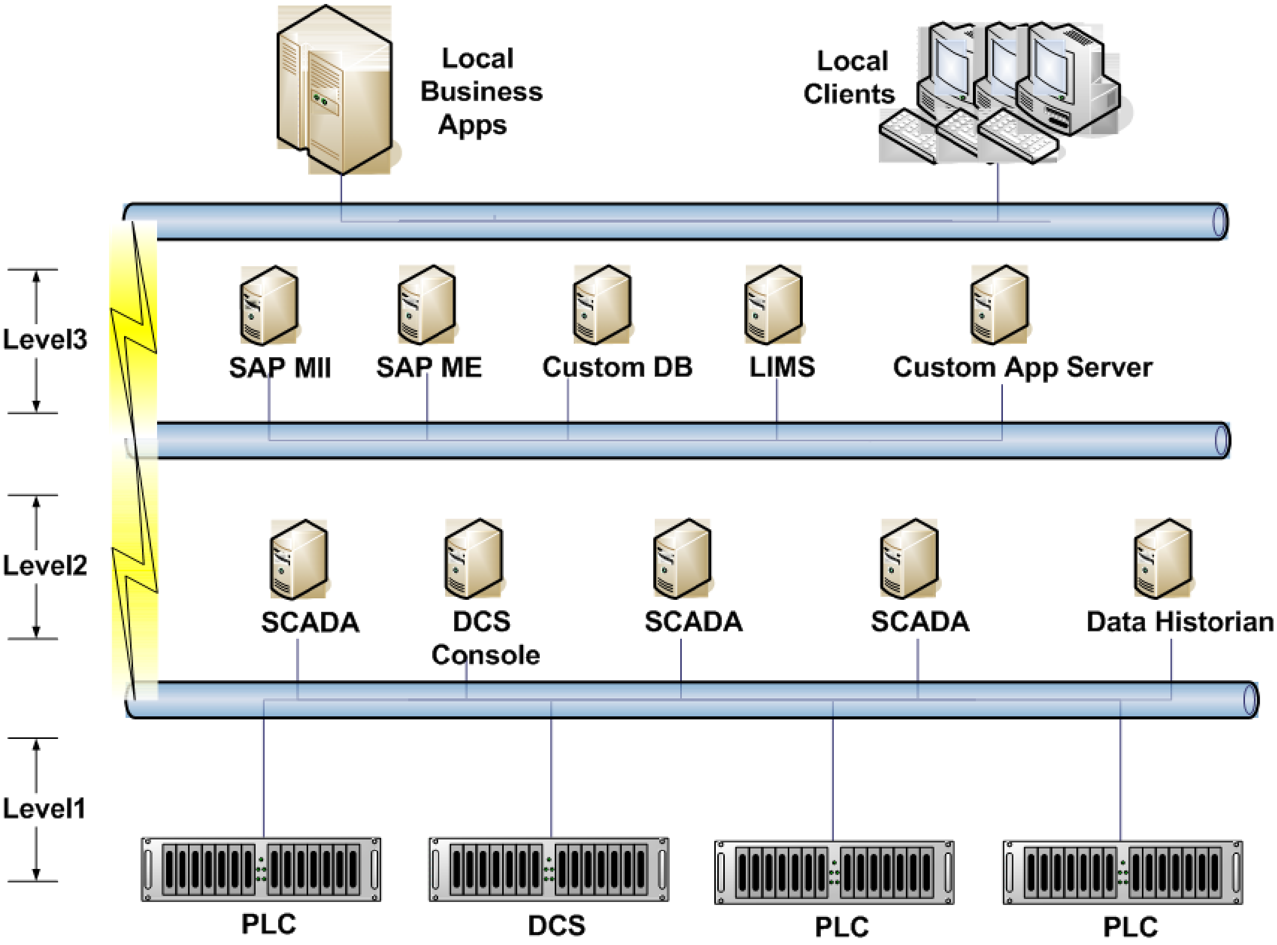

Figure 2 - Typical Manufacturing Environment and Local Manufacutring Systems

The various levels of systems involved and network configurations should be known ahead of your SAP MII implementation in order to best position the product in your environment. While this seems like a pretty straight forward task it is often overlooked and can lead to implementation delays due to network and system access issues. Each system and network/subnet should be addressed individually and assigned to a responsible resource prior to implementation in order to prevent this from occurring. Since there are multiple knowledge domain boundaries that are crossed when performing system integration work, this will often involve an extended team.

Since each company has different systems and then different locations within the same company have different systems with different configurations the SAP MII product is built around connectivity to and generalizing the integration and data returned from these systems. As an example from within the MII product framework there's no need to distinguish between an Oracle and MS SQL server database or the OSI Soft Plant Information and GE Proficy Historians as they return a generic XML data structure. The MII product can then be used to combine, aggregate and return XML data to a user or another system agnostic of the actual system is communicating with. This greatly reduces integration overhead and opens up integration concepts to people whom may not know every detail with the underlying systems but understand the business process.

There is one caveat that must be taken into account during the detailed design phase of your implementation and that is how to manage the thousands of different tags in each of your facilities. Since each of these systems will most likely have a different naming convention and different data recording capabilities the approach in doing so may not be obvious. There, like with all problems, are multiple solutions and depending on your needs you can choose which one is right for you. Each one does involve some sort of lookup or mapping table but there are components of the SAP MII and PCo products that can help you properly maintain this mapping. The most common approach is simply to but the functional name of a tag into a database table and then to create a simple web page to allow the user to interact with and maintain the technical tag names. With the release of MII v12.2 rather than creating a custom database table it is possible to create an MDO object which will maintain this database-like mapping instead of a database table which alleviates database nuances from the implementation. Finally the next approach is to maintain the mapping in the PCo (UDS supports this too) using it in alias mode to define the mapping structure. It is possible to maintain multiple PCo Query agents in a single PCo instance. This allows for one PCo Agent to be setup in alias mode with the standard tag mappings and another agent operating in cache with dynamic meta-data mode to access any other site specific historian tags. This approach seems to work the best as it supports both the standard and custom approach.

Each of the previously mentioned solutions are only proposed ideas and concepts to help to define your application requirements to best fit your company and application needs.

MII Implementation Considerations

The MII Application is a development framework the implementation of the product varies from site to site based on the individual needs of the facility. There are typically common components that can be developed and rolled out across multiple locations and these are important to identify before any application development starts. This will help to limit the amount of application design rework involved when rolling out applications across your business.

In order to properly position the SAP MII product in your landscape it's important to understand each of the primary functions that the product provides. Below is a diagram of the various system connections that are supported by the MII product:

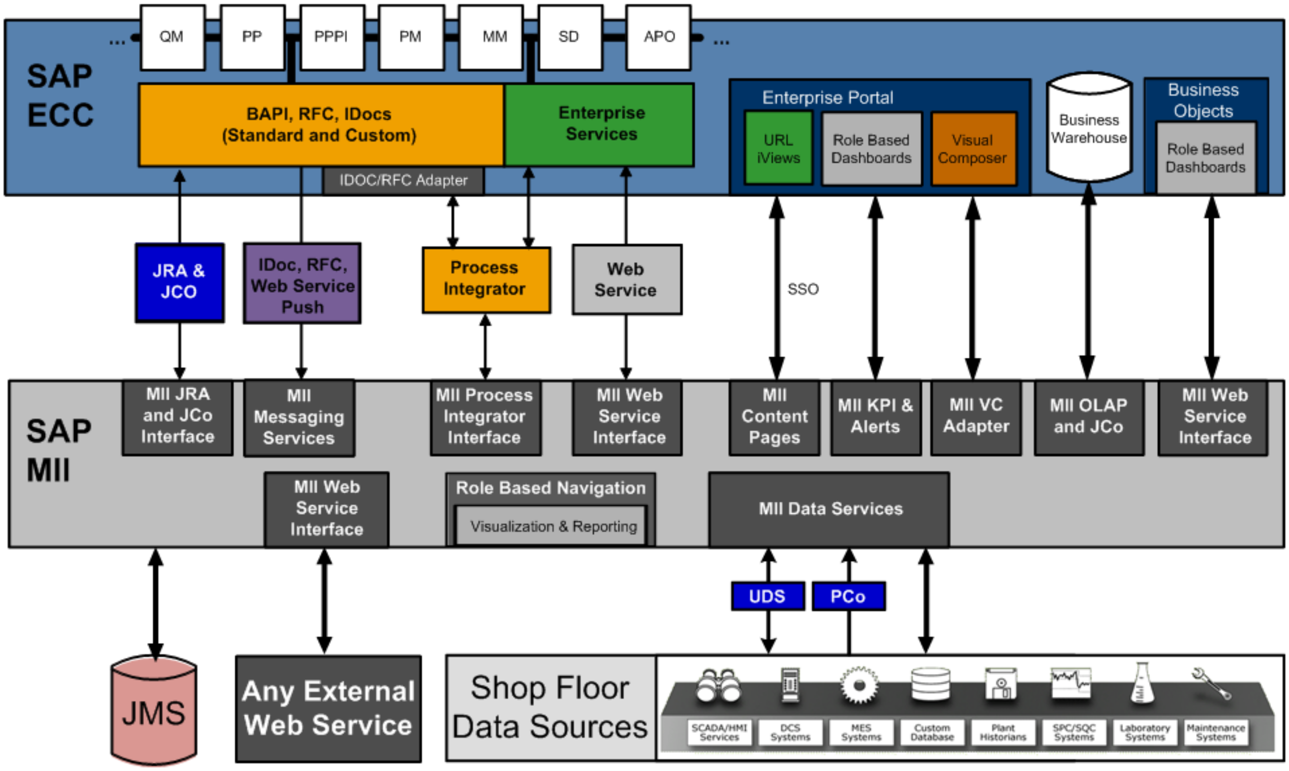

Figure 3 - The SAP MII Integration Landscape

The above diagram can be a bit overwhelming at first so we'll address each integration point can be used in conjunction with each other through the use of the business logic engine or queried back to the user via direct queries through MII Query Templates. The diagram below helps to illustrate the functional architecture of the SAP MII product for accessing and manipulating system data.

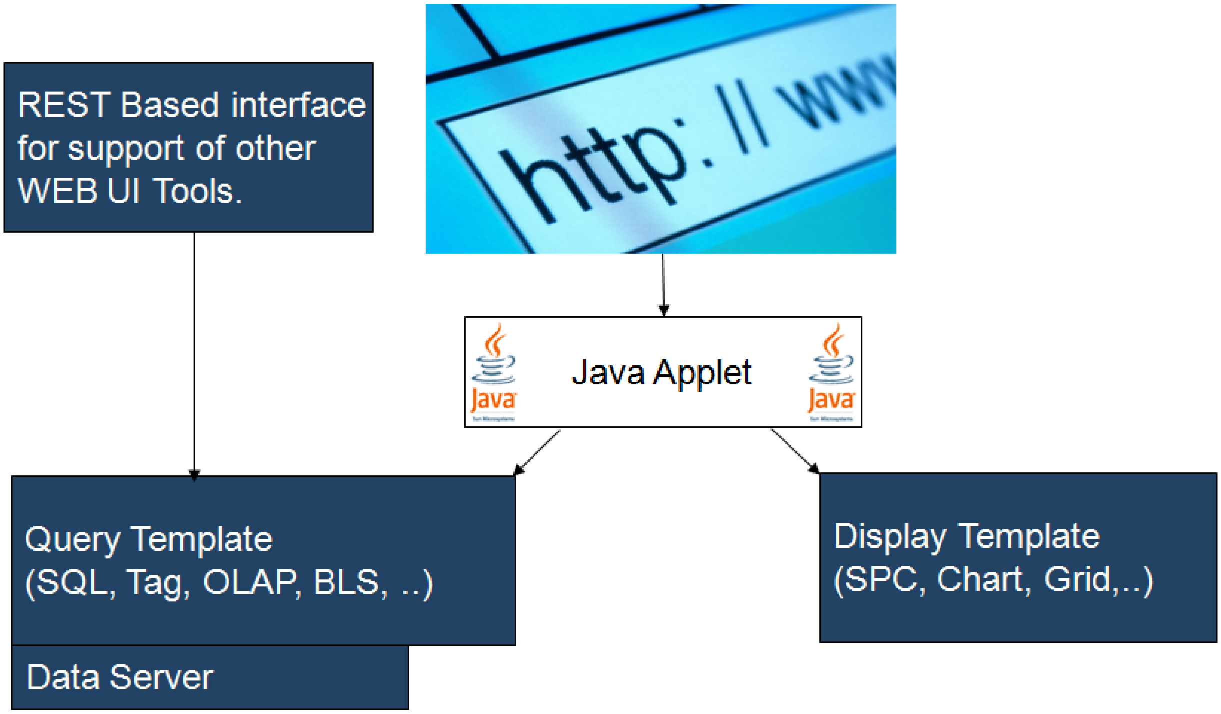
REST Based interface for support of other WEB UI Tools.
http: // www
Java Applet
Java
Java
Query Template (SQL, Tag, OLAP, BLS, ..)
Data Server
Display Template (SPC, Chart, Grid,..)

Figure 4 - High-level Functional Diagram of MII

The above diagram provides a simplified view of the SAP MII product from the perspective of a web page where the standard SAP MII applets or some other web tools can be used for data access and presentation. Since all data that is accessible to the SAP MII product is accessible via URLs and Web Service calls a wide variety of technologies can be used in what is commonly referred to as a "Mash-up". This helps to provide the best end-user experience and leverage existing technical competency within your business. The next diagram provides a more in-depth look at what makes up the data portion of the product and how the various components can be used.

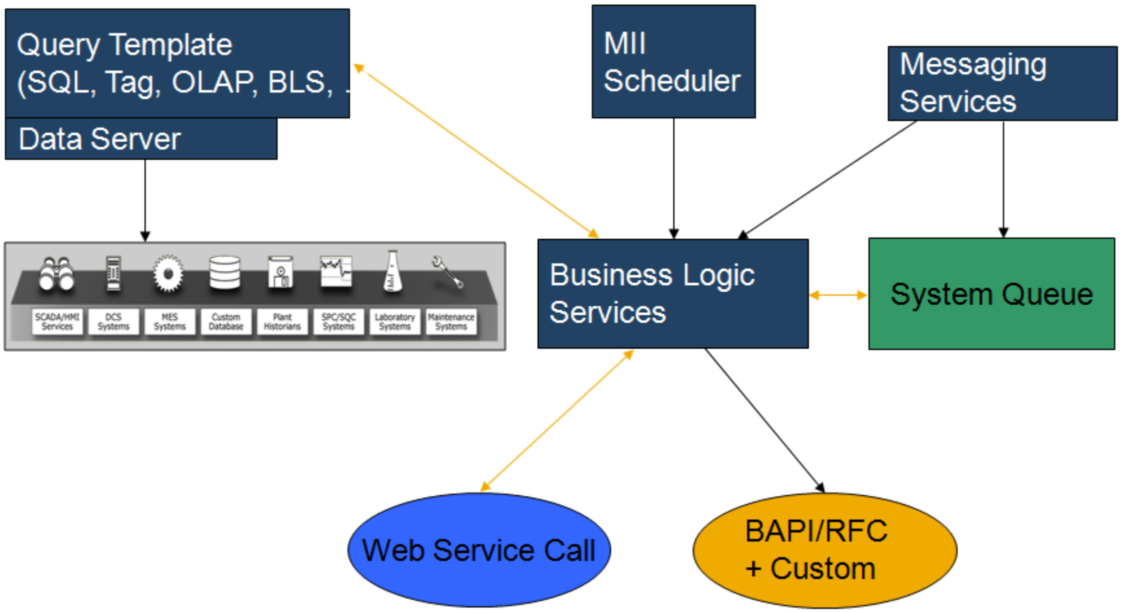

Figure 5 - Overview of the MII Data Services Layer

The SAP MII data services layer provides a wide range of capabilities in order to provide a robust interface for systems integration and data manipulation. The black arrows in the above diagram represent unidirectional interfaces whereas the golden arrows represent bidirectional interfaces. The three golden arrows highlight the ability of the logic engine to be called via a Query template and also to call query templates during their execution for data retrieval. The "Web Service Call" bubble indicates that the logic services and "Call" a Service and also be called as a service and same with interfacing to the SAP MII System Queue. The MII scheduler can trigger events on a time/calendar based requirement and the messaging services call trigger a transaction call upon receipt of a message via the Asynchronous interface. When you put all of these capabilities together you get the image below.

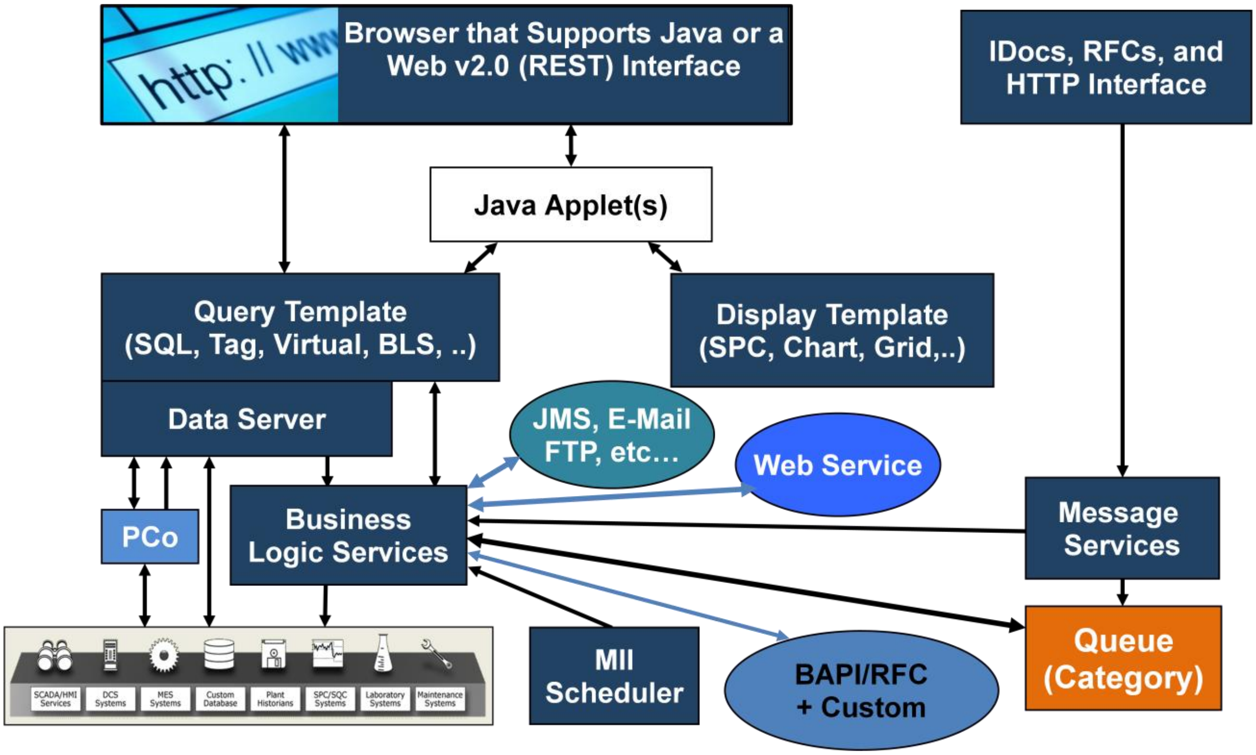

Figure 6 - End to End High-Level view of SAP MII 12.1

The wide range of capabilities provided by the SAP MII tool allow for a simplified and flexible systems integration applications to be quickly developed and maintained.

Configuration Considerations

There are a variety of parameters that can be set on the MII Server that will have an effect on both the throughput of the application and it's footprint on the host system. To reduce the footprint on the host machine, specifically when operating in a production environment the following configuration should be taken into consideration.

First it is important to know about the optimization changes that were made to the MII transaction engine from the 12.0 to the newer releases. This topic includes knowing how the Transaction Manager, Persistence, Caching, and Log Level parameters all affect the execution speed and logging visibility of a transaction when it runs. Each of these parameters is outlined in the help documentation so only a brief description is given here. The Persistence mode of a transaction determines how the execution history of a transaction is stored or "Persisted" in the MII NW Database tables. For example if a transaction is run with Persistence set to "Always" then each sequence and action of the transaction is always stored in the transaction database table. Any entries for a transaction execution are then visible in the Transaction Manager screen and details about its execution are also visible. The granularity level of the execution history is determined by the Log Level defined for the transaction. The typically recommended settings for these parameters in a production environment are setting Persistence to **ONERROR** and Log Level to **Warning**. The rationale behind this is that only if something goes wrong log the issue to the database so that the root issue can be easily identified and corrected. This reduces the overhead for each running transaction on the system to have to write each execution step to the NetWeaver database, improving overall system performance and throughput.

Transaction caching is also something that is new in the post 12.0 releases of MII that allows the application to quickly trigger the execution of transactions. This is done by caching a transaction image in system memory so that it can be used for execution rather than creating a new transaction image each time a transaction is triggered.

Instance Optimization Techniques

Other optimization techniques that can be used for improving various scenarios are to use the follow guidelines; this is in addition to the standard database health monitoring tools:

- Verify that you have the Persistence and Log Levels set properly for the System Administration, Schedule Editor Jobs, and Processing Rule Editor entries.
  - View the “Transaction Manager” screen to identify and address any failed transactions or to identify if you need to properly set the Persistence mode for other transactions.
- If you are using an MII Data Server of type DataSource, set the “NetweaverDataSourceCaching” parameter to true in the System Administration screen.
- If you are using an MII Data Server of type IDBC, UDC or PCoConnector decrease the QueryTimeout parameter from the default of 60 to 20.
- View the “System Jobs” screen and verify that none of the major jobs run at peak load times; rule of thumb is that the less frequent the job, the higher the overhead on the system.
- View the Active Sessions page in order to view potential issues with system based calls that are creating HTTP sessions and modify their interface to include the parameter Session=false in the URL request to the MII server.
- Use the NetWeaver Administrator, Availability and Performance Management -> Resource Monitoring -> Java System Reports page to view system utilization history and peak time periods.

Plant Connectivity

The following section addresses the development and implementation strategy behind the SAP Plant Connectivity software product. It will address specifically why the capabilities are separate from the MII and how it applies to a real-world implementation scenario. It is important to note that the SAP Universal Data Server (UDS) software product and its capabilities are replaced by the SAP Plant Connectivity (PCo) software product, in addition to providing additional functionality.

Plant Connectivity (PCo) Agents

The configuration of the PCo agent instances and communication methods to MII there are a variety of settings that affect overall throughput and performance. When configuring an agent it is important to keep in mind the business use case, this will ensure proper performance and reliability of the instance.

The PCo software has been designed to replace the currently existing UDS product with a more robust and flexible framework in which new agents can be easily developed and added. There are two types of agents which are supported by this product one is for asynchronous notification to MII and the other is for synchronously querying data out of a data source (Previously available via the UDS).

The synchronous agent capabilities are designed to query an OLEDB, OPC, or Native Historian systems and return data to MII in standard Illuminator Document XML format. This allows for the MII product to, in a historian agnostic manner, retrieve data from these systems for processing without needed to know any details about the system. In using this approach each historian system appears the same in the MII environment which helps to eliminate confusion and provides greater application development flexibility.

An asynchronous agent is designed to “subscribe” to a set of tags that exist in a data historian. The agent will monitor the tag(s) values for conditional violations and send an XML message directly to the MII Transaction Engine to process the message. The transaction can update a local event database or utilize some other means to persist the event so that an end user can review the current state of their manufacturing environment. This can be very useful in detecting quality and performance anomalies that can occur during the normal manufacturing process and eliminates the need to poll the data historian which decreases the overall load on the historian system.

PCo 2.2 Side Note

This most recent release of PCo includes additional native agents and also RFC and File Monitoring capabilities. The RFC interface is designed to replace the SAP ODA product for direct integration with the ABAP Based SAP Business Suite and the File Monitor interface helps to improve the integration with legacy systems that require flat-files for communication. You can view the release notes for this product for additional information on its integration capabilities.

PCo Implementation Architectures

There are many ways in which a manufacturing network is designed and the Plant Connectivity product can work across a variety of different scenarios to fit the performance and security needs based on this. The following are three typical installation architectures for the PCo software are shown below:

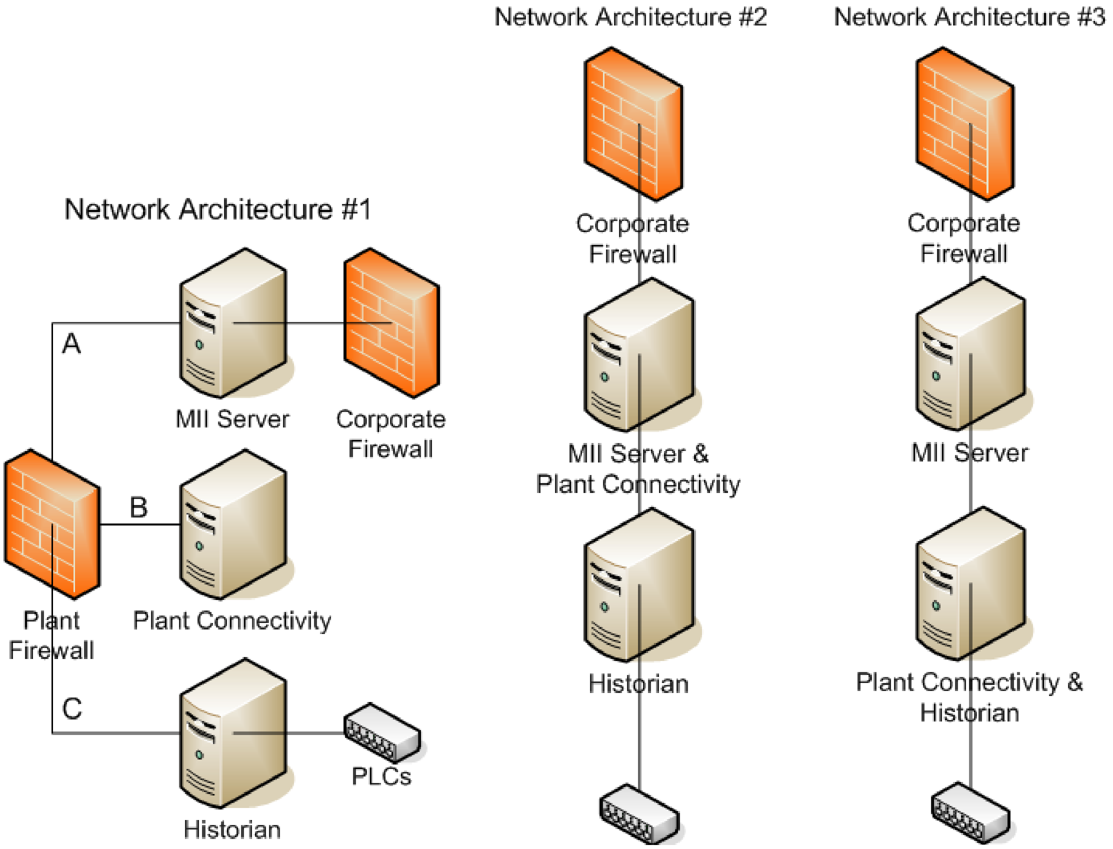

Diagram 1 - Typical Plant Installation Architectures for MII and PCo

Architecture #1

This is the most common scenario, is based on the ability of the network to control what software can and cannot communicate with the Historian subnet. In the above diagram subnet ‘A’ can only communication with subnet ‘B’ and subnet ‘C’ can only communicate with subnet ‘B’. This helps to prevent any system from gaining unauthorized access to the critical shop-floor control system. This installation architecture maintains the necessary network security and still enables the MII Application Server to access the necessary data from the control system at the same. This scenario also allows for each of the three systems to have their own dedicated resources which can be independently sized based on application and user requirements.

Architecture #2

This is the second most common scenario, is based on the fact that the MII server and data historian are on the same subnet but due to validation reasons the PCo software is not installed directly on the Historian server. This limits the implementation costs since the historian system is untouched and will perform better than Architecture #1 because there is no network overhead between the MII and PCo software products.

Architecture #3

This is the best performing scenario, is based on the ability to install the PCo software directly on the historian system. This eliminates the DCOM communication overhead between PCo and the Historian and requires that only the native binary communication between PCo and MII travel across the network. The ability to configure this scenario depends on many factors, most notably the amount of available resources on the Historian system to accommodate both the Historian and PCo loads.

Configuration Considerations

There are a variety of parameters that can be set in the PCo agent that will have an effect on both the throughput of the application and it's footprint on the host system. To reduce the footprint on the host machine, specifically when operating in a production environment the following configuration should be taken into consideration.

The initial startup time and base memory footprint of the PCo agents can be affected by modifying the Cache Mode parameters, which is located on the "Tag Query" tab of the agent configuration. These modes greatly impact the startup and runtime performance of the agents on your system and it is important to understand the implications of each setting to choose the one that best fits your needs. The standard recommendation here is to use **Demand** Mode which will load tag tags and their metadata when queried for the first time and add this information to the internal cache. More information about each of these settings and their effect on the system can be found in the Plant Connectivity Help documentation under the Agent Instance -> Tag Query Tab navigation tree.

The agent should not be responsible to wait and process success or failure execution responses of the message from MII and should only wait for confirmation of message delivery to MII. The reason for this is that if the message processing fails then the MII transaction logic can control the logging and retry required and the PCo agent only needs to log a failed communication message to the MII system. This will prevent potentially long running notification threads that are waiting for an MII transaction response before then can complete which will occur if run synchronously, and as a result each agent should be setup to call a transaction **Asynchronously**. If the MII Application server is too busy to handle an inbound message then the reliability configuration of the agent determines what happens next.

The Persistence model of the transaction also affects how well quickly execution occurs and in a production environment this is recommended to be set to **ONERROR**. This setting causes in the execution of the transaction only being visible in the transaction monitor if there's a processing error and reduces the database interactions between MII and the NetWeaver database and increases performance.

The final setting pertains to the potential overhead of an Agent on the target system and that is the Polling Interval value which affects the dead-band of an event. These settings reside in the Group Settings section of the General tab for some of the Historian connections.

All of the aforementioned settings are important to understand and setup properly in order to reduce the impact on the PCo Host, Source, and Target systems. So be sure to properly configure and maintain each with accordance to your business needs and don't simply rely on the defaults.

Related Content

SAP MII & PCo SDN Forum

SAP MII & PCo SDN Wiki

SAP ME, MII & PCo Help Documentation

SAP Manufacturing Downloads

SAP MES Integration Package for ERP

For more information, visit the Manufacturing homepage

Copyright

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