Tech Note 657
Wonderware Performance 3.5 Software: Performance Tips

Topic#: 002421
OpsManage09 Session#: TS109
Created: November 2009

Introduction

This *Tech Note* details a number of performance-related concepts that must be considered when implementing Performance 3.5 and its Utilization Capability Object (UCO) and Reports.

Implementing the system using these guidelines should improve your system performance for a large database while your database grows in size.

Application Versions

- Wonderware Performance 3.5
- Wonderware Application Server 3.0 and later

OEE Requirements and Calculations

Figure 1 (below) shows the KPI Calculations used for OEE Conditions.

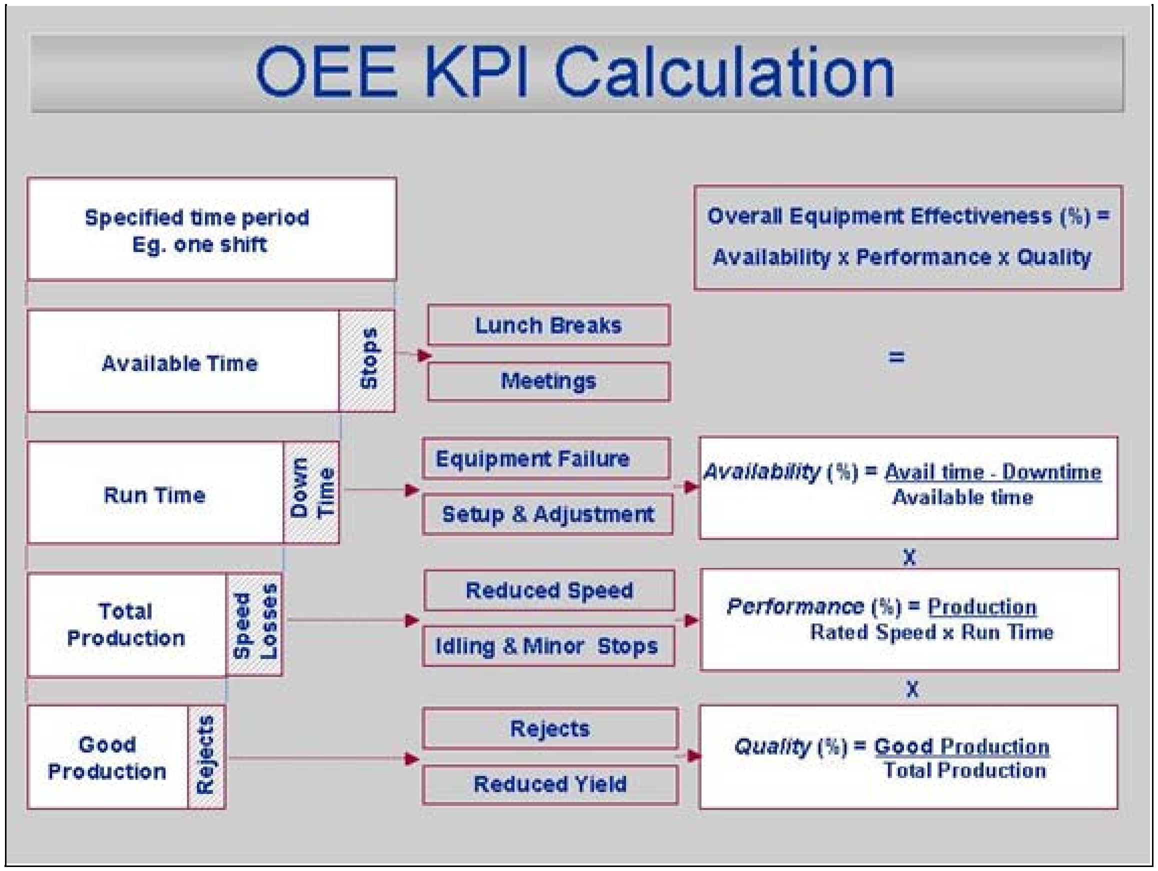
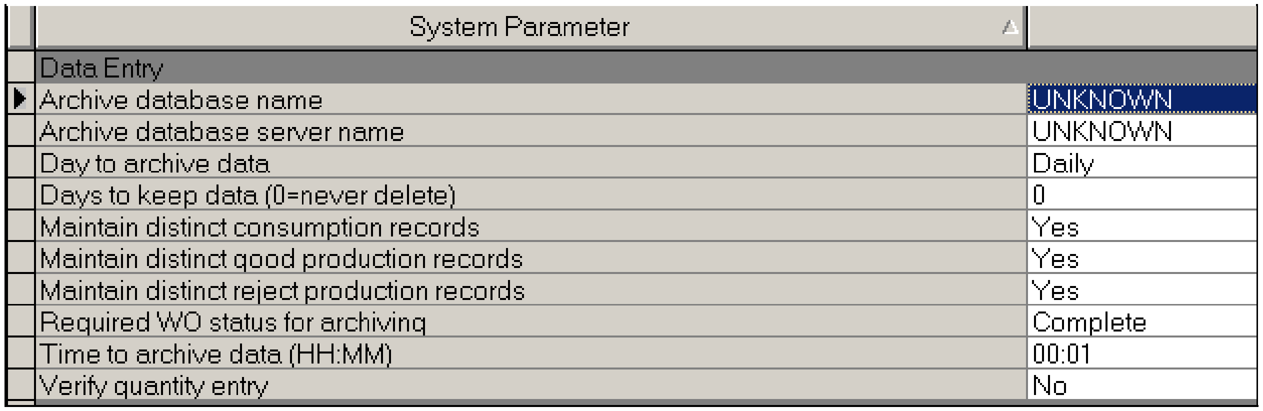


FIGURE 1: OEE CALCULATIONS

OEE Hourly Reporting

By default, Wonderware Performance 3.5 is configured to capture data in order to enable Hourly OEE reporting. Hourly OEE reporting is possible using the following General System Parameters:

- **Maintain Distinct Good Production Records = Yes**
- **Maintain Distinct Reject Production Records = Yes**



System Parameter	
Data Entry	
Archive database name	UNKNOWN
Archive database server name	UNKNOWN
Day to archive data	Daily
Days to keep data (0=never delete)	0
Maintain distinct consumption records	Yes
Maintain distinct good production records	Yes
Maintain distinct reject production records	Yes
Required WO status for archiving	Complete
Time to archive data (HH:MM)	00:01
Verify quantity entry	No

Figure 2: General System Parameters

Having these settings enabled creates additional records in the database, since a new record is created every time production is recorded.

Setting these options to **No** reduces the number of database records stored, but the minimum time period for reporting is **Shift OEE** (not hourly). These settings cause records to be *updated* with *current* production counts where possible, instead of always creating new records.

UCO Modeling and Deployment in Application Server

A UCO must be placed under an Application Object (not System Object, like an Area). The Application Object represents a Performance Entity, the UCO represents a capability of that Entity to capture utilization data. Factelligence Middleware, or Middleware Proxy, must be installed on nodes where Platforms containing UCO objects are deployed.

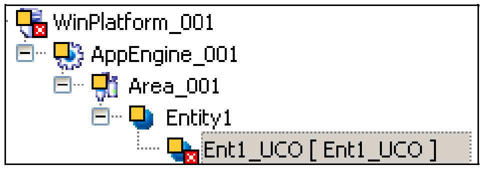
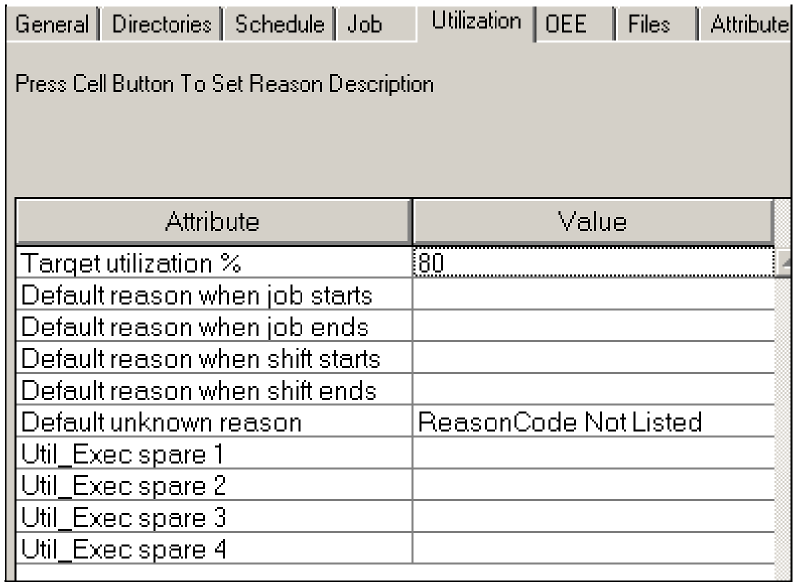


Figure 3: UCO Modeling in Wonderware Application Server

Certain utilization configuration options are only available from the **Utilization** tab in the Configurator application. They can't be set in the UCO.

- **Default Reason when Job Starts**
- **Default Reason when Job Ends**
- **Default Reason when Shift Starts**
- **Default Reason when Shift Ends**



Press Cell Button To Set Reason Description

Attribute	Value
Target utilization %	80
Default reason when job starts	
Default reason when job ends	
Default reason when shift starts	
Default reason when shift ends	
Default unknown reason	ReasonCode Not Listed
Util_Exec spare 1	
Util_Exec spare 2	
Util_Exec spare 3	
Util_Exec spare 4	

FIGURE 4: PERFORMANCE CONFIGURATOR CLIENT UTILIZATION CONFIGURATION

UCO Extensibility Considerations

This section outlines the behavior of certain object-specific attributes of the Performance Utilization Capability Object (UCO), and the limitations of those attributes in the context of writing data changes to the Performance database.

The Performance UCO has object-specific attributes that can be extended.

The UCO communicates with the Performance database via the Wonderware Factelligence COM+ Middleware application. It is possible to configure the UCO attributes in an incorrect manner, causing the UCO to attempt database writes via Middleware on every scan of the UCO.

Incorrect configuration can cause CPU usage on the platform hosting the UCO to become unacceptably high, resulting in errors and/or non-responsive Middleware. It can also overload the database server, causing SQL deadlocks, etc.

Attribute Extension Considerations

The following attributes should *never* be extended to an input source listed on the **Extensions** tab of the UCO object.

Doing so results in writes to the Performance database on every scan, resulting in an extremely large number of database records.

Changes to these attributes should be controlled via scripting.

Trigger Attributes

- CounterX.AddProdQtyTrg
- ProdAttrs.StartJobCmd
- ProdAttrs.EndJobCmd

OEE Performance Targets

- TargetOEEPercent
- TargetPerformancePercent
- TargetQualityPercent
- TargetUtilizationPercent

Rollover Production Counter

CounterX.AddProdQtyCntr: This is an exception to the input source restriction, but only under certain conditions.

- If this is extended to an input source that updates faster than the UCO scan, a database write will occur every scan, IF both Deadband and Update Interval are zero for the counter on the Production Counters tab.
- You can configure a Deadband or Update Interval for the counter to make the database writes less frequently. If the input source is slow to update (for example you make 4 pieces an hour), this attribute can be extended without problems.

UCO Event Skipping

Performance configuration requires the designer to understand the customer environment, and in particular, its micro-stoppages dynamics.

Each installation needs to determine where micro-stoppages should be categorized.

- In **Availability rate** or
- In **Performance rate**

The following scenario illustrates this requirement.

- Total Production Time: **100 minutes**
- Equipment Rate (nominal): **90 units-per-minute**
- Accumulated micro-stoppages (during production time): **10 minutes**
- Total Produced Units (all good units): **8100**

Scenario where Micro-Stoppages	Up	Down	Operation Time	Total Production Time	Availability Rate	Performance Rate	OEE %
Account for Uptime	90	10	90	100	90%	100%	**90%**
Account for Downtime	100	0	100	100	100%	90%	**90%**

Figure 5: Micro-stoppages Matrix

Customer management Policy must establish which metric the Micro-stoppages will affect.

- **Downtime**: Which means Micro-stoppages impact Availability Rate.
  - Maintenance and Engineering functions are most impacted by this OEE component.
- **Uptime**: Which means Micro-stoppages impact Performance Rate.
  - The Operations function is most impacted by this OEE component.

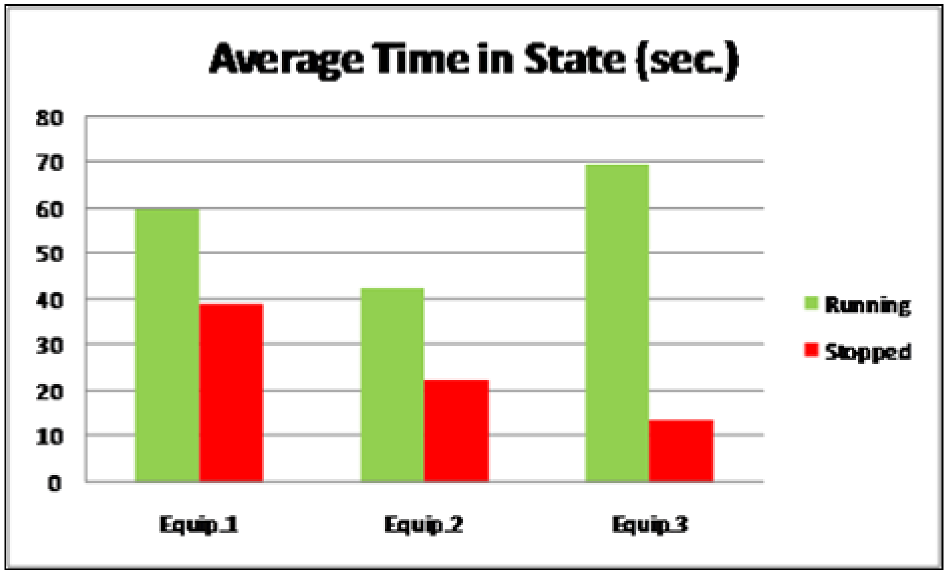
A general rule is to discover how long it takes for the maintenance/engineering crew to react to downtime events. For example, if maintenance and engineering don't want to be called for events lasting under 5 or 10 minutes, then Micro-stoppages should largely be operation-related events, and therefore affect **Performance rate**.

On the contrary, if maintenance and engineering are able to react to events lasting under 2 or 3 minutes, then Micro-stoppages should impact the **Availability rate**.

Understanding Equipment Event Occurrences

Performance historizes the Equipment states, the majority of which are **Running** and **Stopped**. Depending on packaging line characteristics, the trend analysis can look like the following graphics.

Note: This data should be a part of *any* project assessment.



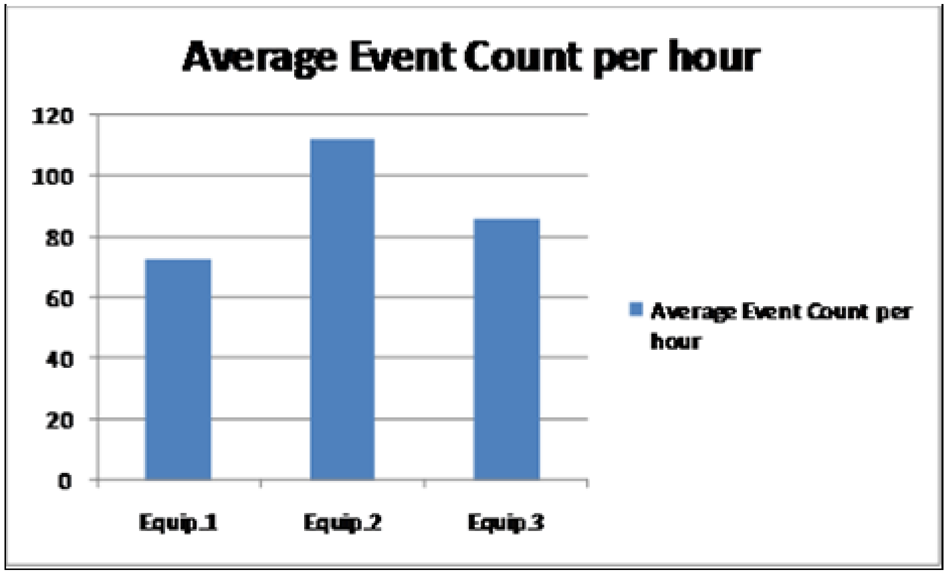


FIGURE 6: EQUIPMENT STATES AND EVENTS ANALYSIS

- Average Time in State: Determines running/stopped average duration-per-equipment.
- Average Event count-per-hour: Determines average number of unique records created.

The percent cumulative Occurrences shown in the following graphic helps to understand when most the equipments' event duration occurs, and hence helps the designer understand where to set the threshold for raw reason code activation time.

This is important in order not to exceed the core product (Performance 3.5) capability. This helps to understand when to set the Activation time for micro/minor-stoppages. Figure 7 (below) shows that **Equipment 1's** Activation time could be slower than **Equipment 2's**.

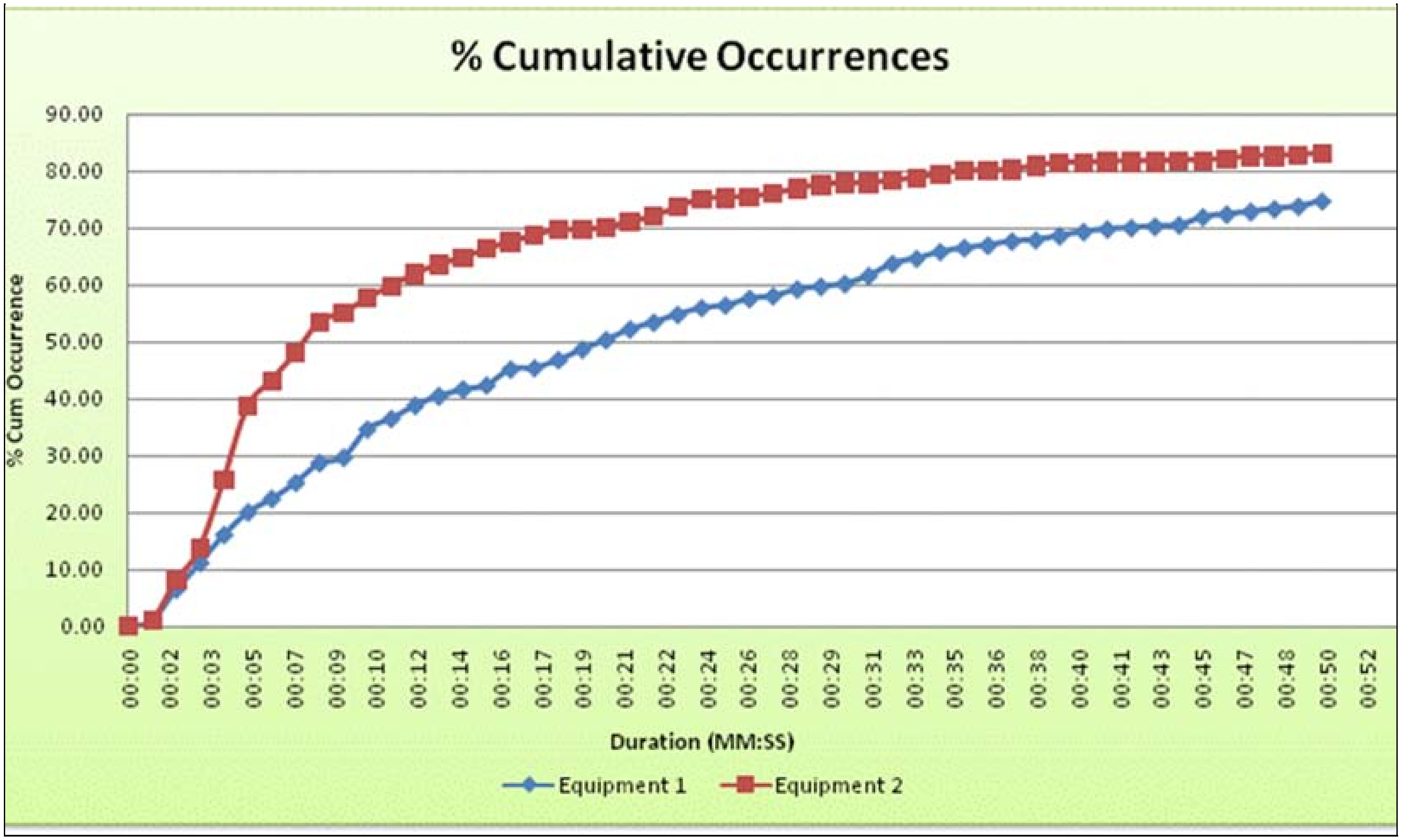


Figure 7: Cumulative Occurrences

The percent contribution of total downtime duration (Figure 8 below) helps to isolate the downtime event duration with respect to its contribution to downtime.

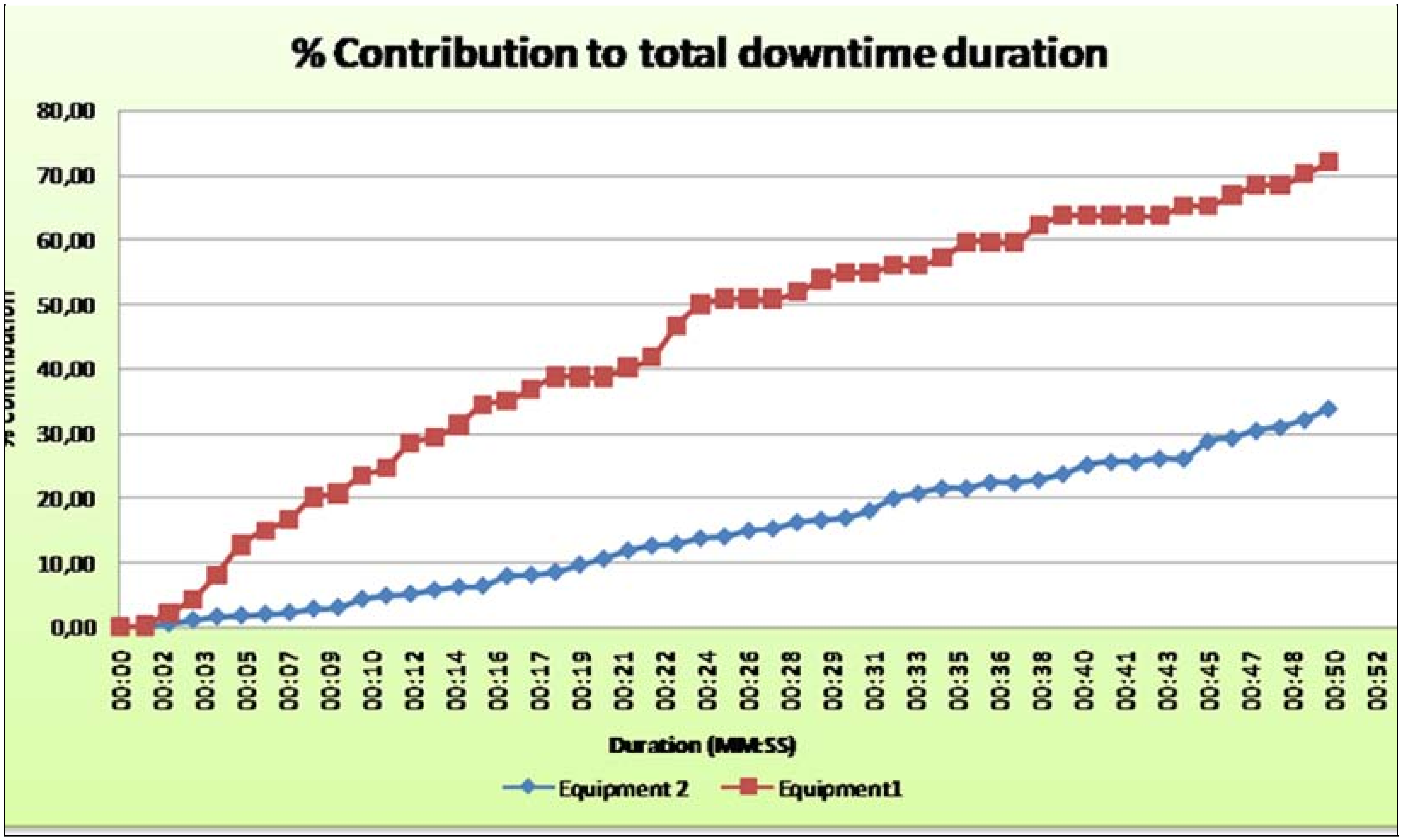


Figure 8: Contribution to Total Downtime

Why the Proper Activation Time Setting is So Important

If the equipment cycles too fast (event duration too short), Performance will miss the event (either uptime or downtime events). Missing events propagates into calculations like **Availability** and **Performance** Rates.

Furthermore, fast-cycling events also translate into bigger databases. Bigger databases will require more time to produce reports or retrieve data for .NET controls. In order to maximize the value and experience with the customer, event durations and frequency must be factored into any design.

Setting Performance Threshold for Micro- and/or Minor Stoppages

Event activation time configuration determines the floor, or threshold event duration has when configuring Performance 3.5.

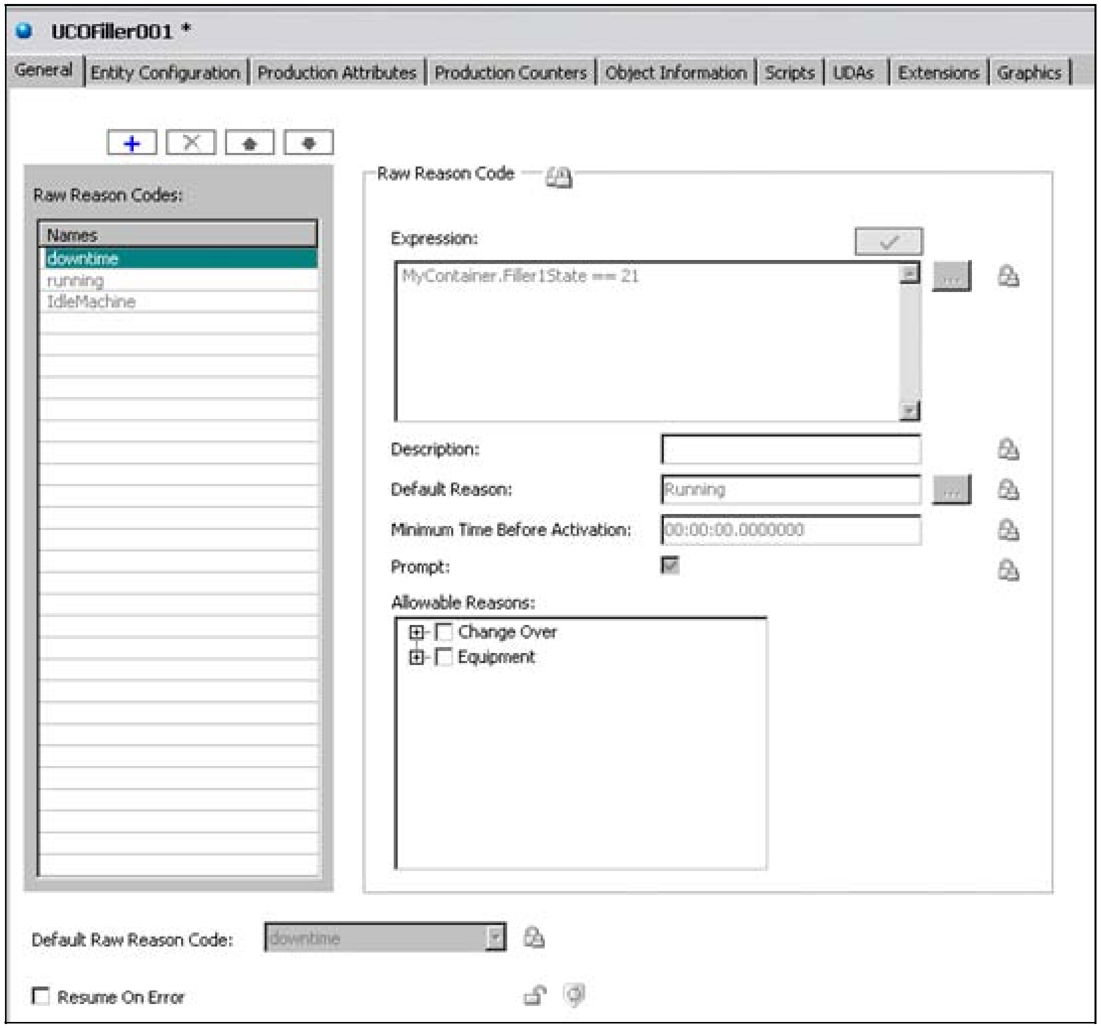
- **Detailed Raw Reason Codes/RRCs** should be set between 1-5 min. (1st to evaluate on UCO object)
- **General states** should also be set between 1-5 min. (2nd group to evaluate after RRCs)

A reduced-speed, single RRC at the bottom of the UCO RRC list (or before the Unknown RRC) should be set to **0** seconds in order to avoid event cycling. This RRC will capture all events lasting between 0 and 1-5 minutes and insert one record into the database showing Reduced Speed.

Based on the Customer Management Policy, it will determine if it wants it recorded as **Uptime** or **Downtime** loss.

Setting Minimum Time Before Activation in the UCO

Setting up the Minimum duration in the UCO is done in the **General** tab and is defined for each of the specified Raw reason Codes.



Expression: MyContainer.Filler1State == 21

Description:

Default Reason: Running

Minimum Time Before Activation: 00:00:00.0000000

Prompt: ✓

Allowable Reasons: Change Over, Equipment

Default Raw Reason Code: downtime

Figure 9: UCO General Settings

Notes

- Activation time should never be set under 60 seconds.

Activation time for all Detailed Downtime and Equipment States should always be the same value.

- If the customer wants to get full granularity into ALL event durations (for those between 0 and 1-5 minutes), it is recommended to historize the PLC individual downtime Boolean tags (not included in Performance).

Summary

Before you begin configuring Performance, you should do the following:

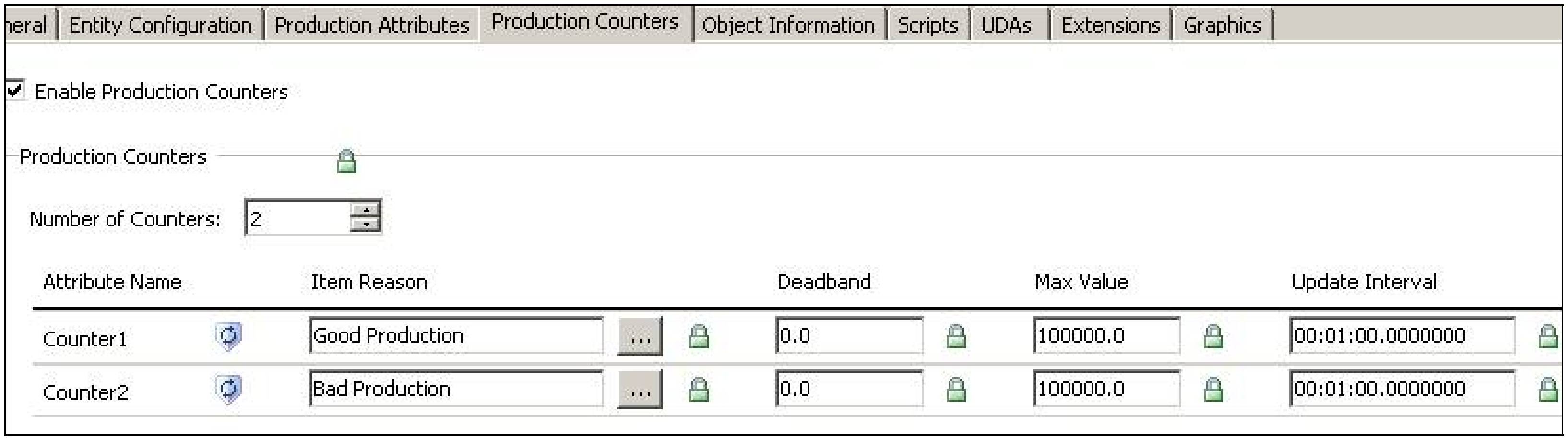
1. Assess historical and current packaging line characteristics with the customer.
2. Historize (Historian) the Equipment State tag during typical production conditions.
3. Analyze results (in particular Average Time in State, and event Frequency over the same period.
4. Define preliminary system topology based on number of lines and equipment.
5. Determine the size of the database and the archiving strategy, in order to sustain system performance over time.
6. Determine which threshold needs to be set based on 1.,2.,3., above.
7. Configure the threshold in Performance object templates.

Note: When using the Utilization .NET control for manual Utilization changes, it is important to know that the grid is populated from the **util_log** and the **tpm_stat** tables.

When these tables get very large, the initial loading of the control can take a long time. This issue has been reported to Wonderware Support and Development. Hot Fixes are available for this issue and should be requested through **Wonderware Support**.

Production Counters from the UCO

You can define up to 10 rollover counters in a single UCO object. To use the rollover counter capability, click the **Enable Production Counters** option in the UCO object and use the correct extension attributes.



☑ Enable Production Counters

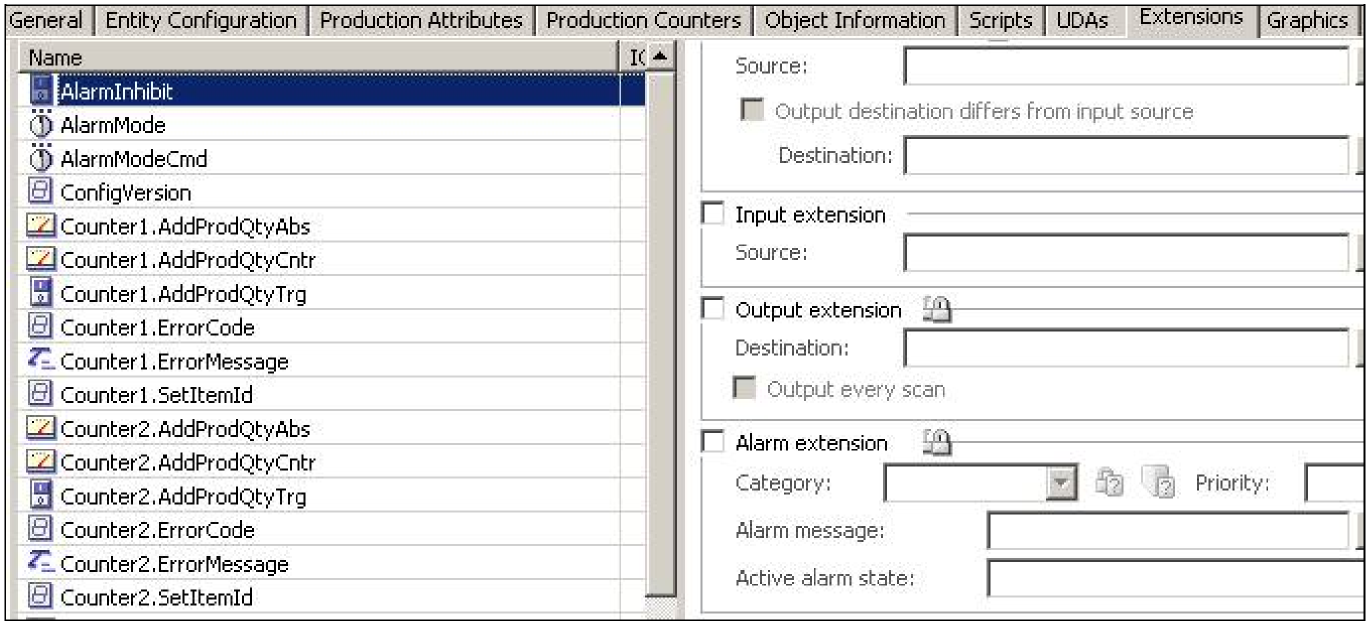
Production Counters

Number of Counters: 2

Attribute Name	Item Reason	Deadband	Max Value	Update Interval
Counter1	Good Production	0.0	100000.0	00:01:00.0000000
Counter2	Bad Production	0.0	100000.0	00:01:00.0000000

Figure 10: Utilization Capability Object Production Counters

When you enable Production Counters, a set of Extension Attributes related to those counters is created.



Name

- AlarmInhibit
- AlarmMode
- AlarmModeCmd
- ConfigVersion
- Counter1.AddProdQtyAbs
- Counter1.AddProdQtyCntr
- Counter1.AddProdQtyTrg
- Counter1.ErrorCode
- Counter1.ErrorMessage
- Counter1.SetItemId
- Counter2.AddProdQtyAbs
- Counter2.AddProdQtyCntr
- Counter2.AddProdQtyTrg
- Counter2.ErrorCode
- Counter2.ErrorMessage
- Counter2.SetItemId

Source:

Output destination differs from input source

Destination:

Input extension

Source:

Output extension

Destination:

Output every scan

Alarm extension

Category:

Priority:

Alarm message:

Active alarm state:

Figure 11: Utilization Capability Object Production Counter Extension Attributes

The **CounterX.AddProdQtyAbs** and **CounterX.AddProdQtyTrg** attributes work as a pair and are referred to as an **Absolute** counter. These counters can obtain production data from something like a scale.

The **CounterX.AddProdQtyAbs** value should be extended to a field device providing the measured weight and **CounterX.AddProdQtyTrg** should be extended to a button on the Operator panel that is pressed when the load is stabilized. Pressing this button records the weight to the database.

The **CounterX.AddProdQtyCntr** is a rollover counter. This attribute can be extended to something like a photo eye that just keeps a running count of items that pass by. Since counting cannot go on to infinity in the field device, you need to configure a point where the counter resets to zero. The maximum value in the field device is also configured in the UCO Production Counter's **Max Value** field.

This maximum value needs to be high enough to ensure that the maximum would never be reached twice between UCO updates.

Utilization Capability Object Default Rollover Counter Behavior

A rollover counter is a typically extended to a field device that provides an incrementing value to track material production. The field device is configured with an internal maximum value, that when reached, will roll back to zero. The value should ONLY roll back to zero when the maximum value has been reached. If you have a need to reset to zero per day, shift, work order, etc. you cannot extend the rollover counter attribute directly to your field device counter. You will either need to develop a secondary field device counter that accumulates from the actual counter or script for this behavior in Application Server and extend the counter to a UDA or write to it via a script.

This example provides example scenarios, and assumes a PLC provides the system with production counter information as follows:

```
Initial Value = 0; Max (Rollover) Value = 100000
```

The following scenarios explain PLC counter interactions with the UCO. In this case, when the UCO Object is undeployed, it initializes the counters to 0 on redeploy. This behavior causes undesirable over-reporting of production quantities.

Scenario 1

- Deploy UCO – Counter is 0.
- PLC counter increments to 20000 and updates the UCO counter to 20000.
- 20000 pieces are recorded to the database. (Current value - initial value).
- The PLC counter increments to 67000, updating the UCO counter to 67000.
- 47000 pieces are recorded to the database. (Current value – last value).
- The PLC counter increments to 100000 rolls back to 0 and increments additionally to 4000 for the next read, updating the UCO counter to 4000. 37000 pieces are recorded in the database. (Rollover max – last value + current value). The system will assume the rollover condition occurred since the current value is less than the previous value.
- PLC counter increments to 30000, updating the UCO counter to 30000. 26000 pieces are recorded in the database.
- Undeploy the UCO.
- PLC increments to 35000.
- Redeploy UCO. The counter initializes to 0 on startup. It sees the PLC counter as 35000 and records 35000 pieces to the database. It should have only recorded the difference of 5000.
- Now production is overstated by 30000 pieces.

Scenario 2

- Deploy UCO – Counter is 0.
- PLC counter increments to 20000 and updates the UCO counter to 20000.
- 20000 pieces are recorded in the database. (Current value - initial value).
- PLC counter increments to 67000, updating the UCO counter to 67000.
- 47000 pieces are recorded in the database. (Current value – last value).
- PLC counter increments to 100000, rolls back to 0 and increments additionally to 4000, updating the UCO counter to 4000.
- 37000 pieces are recorded in the database. (Rollover max – last value + current value). The system assumes the rollover condition occurred since the current value is less than the previous value.
- PLC counter increments to 30000, updating the UCO counter to 30000. 26000 pieces are recorded in the database.
- Undeploy the UCO.
- PLC stays at 30000.
- Redeploy UCO. The counter initializes to 0 on startup. It sees the PLC counter as 30000 and records 30000 pieces to the database.

It should not have recorded any production since the PLC counter did not change. Now production is overstated by 30000 pieces.

Utilization Capability Object Rollover Counter Behavior After Installing Hotfix 1394

Hotfix 1394 will help with the rollover counter behavior, causing the UCO to simply miss production while it is undeployed, instead of over reporting. By default, the counter will initialize to the current PLC value instead of 0. You will simply need to manually track and enter what was missed. The fix also delays the counting functionality until the 2nd scan, so you have the opportunity to initialize the counter to whatever you want instead of a forced 0 via scripting, etc.

Scenario 1

- Deploy UCO – Counter is 0.
- PLC counter increments to 20000, updating the UCO counter to 20000.
- 20000 pieces are recorded in the database. (Current value - initial value).
- PLC counter increments to 67000 and updates the UCO counter to 67000. 47000 pieces are recorded in the database. (Current value – last value).
- PLC counter increments to 100000 rolls back to 0 and increments additionally to 4000, updating the UCO counter to 4000. 37000 pieces are recorded in the database. (Rollover max – last value + current value). The system will assume the rollover condition occurred since the current value is less than the previous value.
- PLC counter increments to 30000, updating the UCO counter to 30000. 26000 pieces are recorded in the database.
- Undeploy the UCO.
- PLC increments to 35000.
- Redeploy UCO. The counter initializes to 35000 (the current PLC value) on startup. It sees the PLC counter current value as 35000 and records 0 pieces to the database. It should have recorded the difference of 5000, but has no way to know what the last counter value was.

  Now production is understated by 5000 pieces and this amount can be manually entered.

Scenario 2

- Deploy UCO – Counter is 0.
- PLC counter increments to 20000 and updates the UCO counter to 20000.
- 20000 pieces are recorded in the database. (Current value - initial value).
- PLC counter increments to 67000, updating the UCO counter to 67000. 47000 pieces are recorded in the database. (Current value – last value).
- PLC counter increments to 100000 rolls back to 0 and increments additionally to 4000, updating the UCO counter to 4000. 37000 pieces are recorded in the database. (Rollover max – last value + current value). The system will assume the rollover condition occurred since the current value is less than the previous value.

PLC counter increments to 30000, updating the UCO counter to 30000. 26000 pieces are recorded in the database.

- Undeploy the UCO.
- PLC stays at 30000.
- Redeploy UCO. The counter initializes to 30000 on startup. It sees the PLC counter current value as 30000 and records 0 pieces to the database. Recorded production is now accurate.

Performance Reports

This section explains best practices for generating Performance Reports.

Improving Report Generation Time

When generating reports across linked servers in the Operations & Performance environment, two factors can impact timely report generation:

- The number of Performance reports you generate.
- The SQL Query used to display the report from Wonderware Information Server.

When a report is generated it creates an item in the database. The number of report items increases over time, so report query return times will become longer and longer. In many cases the query times out before any information is returned.

Figure 12 (below) shows the number of report items combined with the query turnaround times in seconds (vertical axis). 200 seconds represents a timeout.

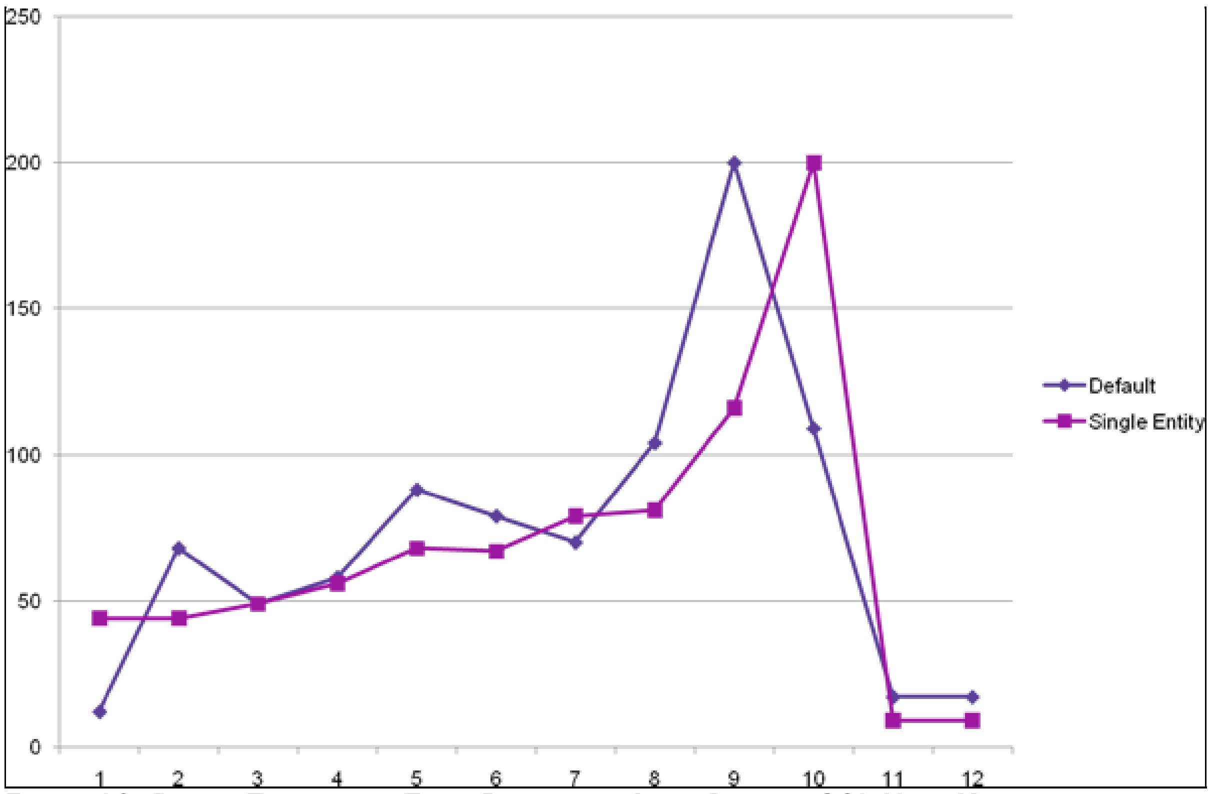


Figure 12: Report Turnaround Times Before and After Dynamic SQL View Modification

The last data points (#11 and 12) represent the turnaround time after modifying the query as described in this section of the Tech Note.

This section outlines how to improve the turnaround performance for the supplied Performance 3.5 reports.

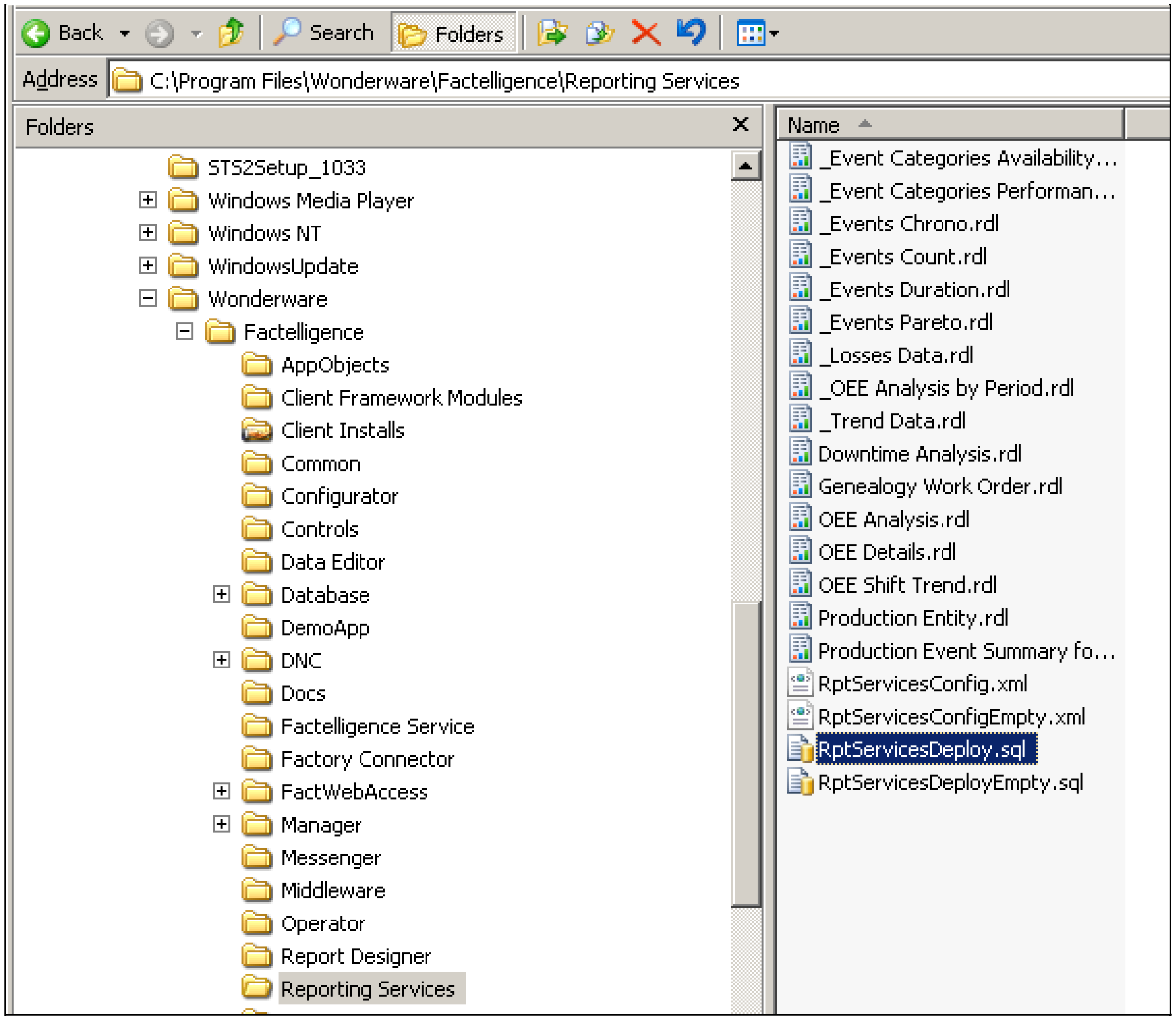
Configuring and Deploying the Report File

To improve turnaround time, you must modify the SQL script that creates the Dynamic Views used by Microsoft Reporting Services. WIS uses the Reporting Services to produce the report.

The SQL script is installed in the **RptServicesDeploy.sql** file. The default installation puts the file at: **C:\Program Files\Wonderware\Factelligence\Reporting Services**.

To modify the RptServicesDeploy.sql file

1. Locate and double-click the **RptServicesDeploy.sql** file.



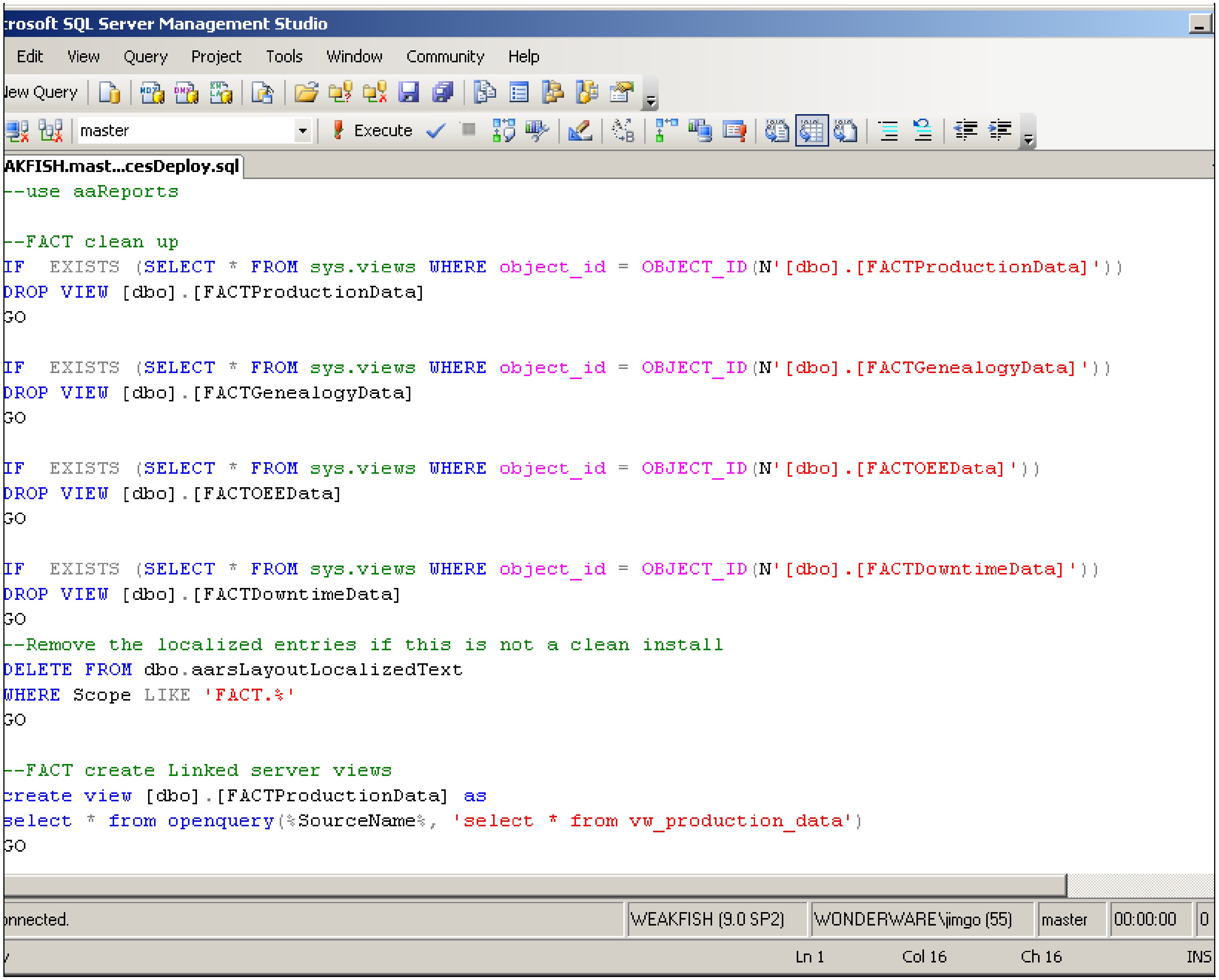
C:\Program Files\Wonderware\Factelligence\Reporting Services

STS2Setup_1033
Windows Media Player
Windows NT
WindowsUpdate
Wonderware
Factelligence
AppObjects
Client Framework Modules
Client Installs
Common
Configurator
Controls
Data Editor
Database
DemoApp
DNC
Docs
Factelligence Service
Factory Connector
FactWebAccess
Manager
Messenger
Middleware
Operator
Report Designer
Reporting Services

_Event Categories Availability...
_Event Categories Performan...
_Events Chrono.rdl
_Events Count.rdl
_Events Duration.rdl
_Events Pareto.rdl
_Losses Data.rdl
_OEE Analysis by Period.rdl
_Trend Data.rdl
Downtime Analysis.rdl
Genealogy Work Order.rdl
OEE Analysis.rdl
OEE Details.rdl
OEE Shift Trend.rdl
Production Entity.rdl
Production Event Summary fo...
RptServicesConfig.xml
RptServicesConfigEmpty.xml
RptServicesDeploy.sql
RptServicesDeployEmpty.sql

FIGURE 13: RPTSERVICESDEPLOY.SQL FILE LOCATION

The SQL Server Management Studio Query window appears. You can also open this file and edit it using Notepad.


rosoft SQL Server Management Studio
Edit View Query Project Tools Window Community Help
lew Query
master
Execute
AKFISH.mast...cesDeploy.sql

```
--use aaReports

--FACT clean up
IF  EXISTS (SELECT * FROM sys.views WHERE object_id = OBJECT_ID(N'[dbo].[FACTProductionData]'))
DROP VIEW [dbo].[FACTProductionData]
GO

IF  EXISTS (SELECT * FROM sys.views WHERE object_id = OBJECT_ID(N'[dbo].[FACTGenealogyData]'))
DROP VIEW [dbo].[FACTGenealogyData]
GO

IF  EXISTS (SELECT * FROM sys.views WHERE object_id = OBJECT_ID(N'[dbo].[FACTOEEData]'))
DROP VIEW [dbo].[FACTOEEData]
GO

IF  EXISTS (SELECT * FROM sys.views WHERE object_id = OBJECT_ID(N'[dbo].[FACTDowntimeData]'))
DROP VIEW [dbo].[FACTDowntimeData]
GO
--Remove the localized entries if this is not a clean install
DELETE FROM dbo.aarsLayoutLocalizedText
WHERE Scope LIKE 'FACT.%'
GO

--FACT create Linked server views
create view [dbo].[FACTProductionData] as
select * from openquery(%SourceName%, 'select * from vw_production_data')
GO
```

Figure 14: MS SQL Server Management Studio Query Editor

2. Use **Ctrl+F** to locate **CREATE VIEW**.

   Instead of using OpenQuery SQL syntax to create the views in the aaReports database, the modification uses a four-part syntax for the creation of the dynamic views.

Note: Some elements of the query are changed to upper-case for clarity in this Tech Note. The query is case-insensitive.

The following query script is provided for clarity:

Replace (old version):

```
CREATE VIEW [dbo].[rpt_vw_FACT_Product] AS
SELECT * from OPENQUERY (aaFactelligence, 'SELECT * FROM vw_product')
GO
```

With (new optimal way):

```
CREATE VIEW [dbo].[rpt_vw_FACT_Product] AS
SELECT * from %SourceName%.Factelligence.dbo.vw_product
GO
```

3. In the four-part syntax, replace **Factelligence** with your database name.

4. Use **Ctrl+F** to locate and change each **Create View** instance in the SQL query:

   Replace:

```
CREATE VIEW [dbo].[rpt_vw_FACT_Entity]
AS
SELECT * from OPENQUERY(aaFactelligence, 'SELECT * from vw_ent_for_oee union select * from vw_ent_for_util')
GO
```

   With:

```
CREATE VIEW [dbo].[rpt_vw_FACT_Entity]
AS
SELECT * FROM %SourceName%.Factelligence.dbo.vw_ent_for_oee UNION
SELECT * FROM %SourceName%.Factelligence.dbo.vw_ent_for_util
GO
```

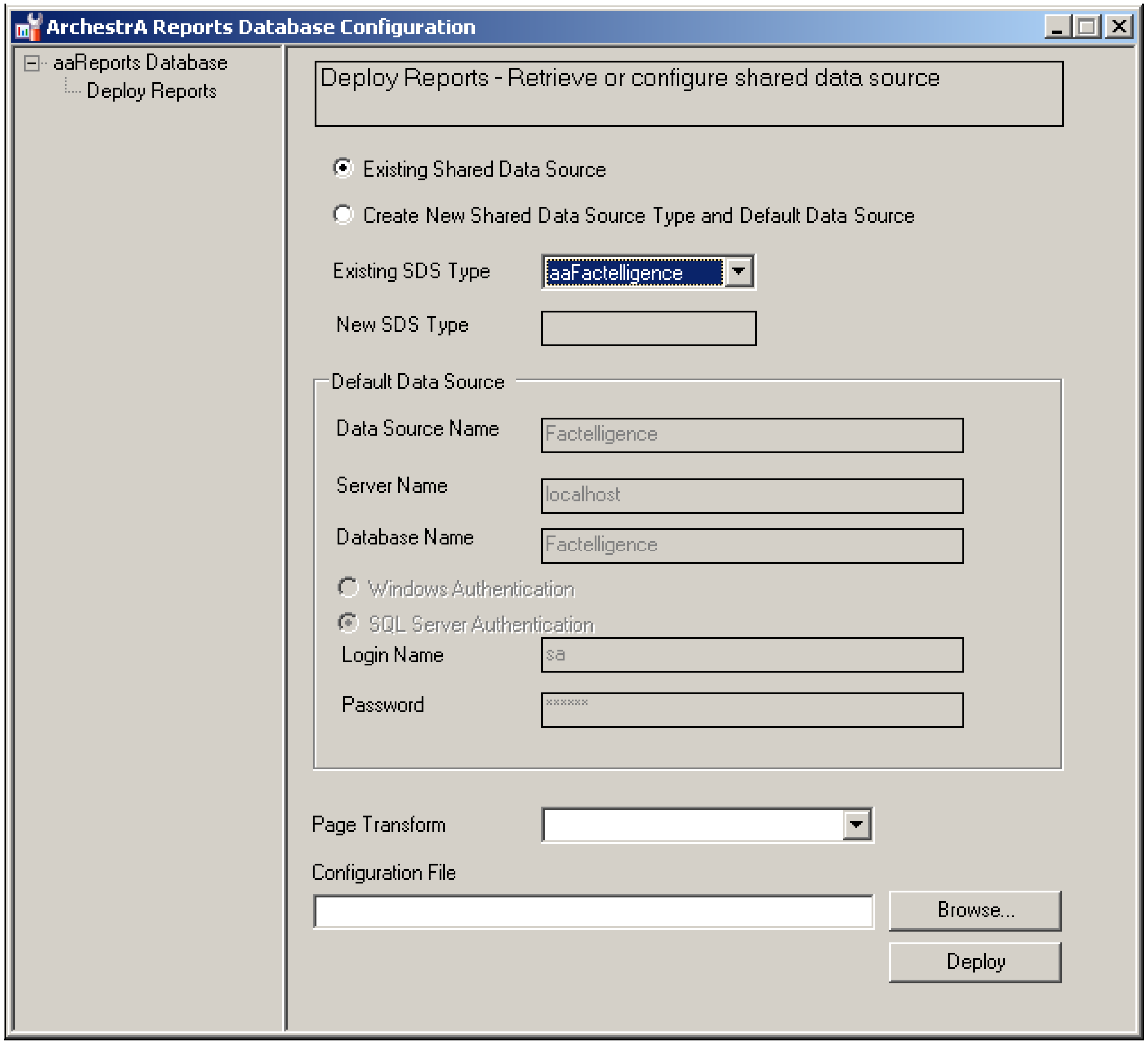
5. Continue changing the syntax in *each instance* of CREATE VIEW. There are 13 in all, including the above examples.

6. Save your changes and close the Query Editor.

   DO NOT execute the query from within SQL Server Management Studio.

7. On your Wonderware Information Server node, run the **aaReportDBConfig** utility using **Start > All Programs > Wonderware > Information Server > Deploy ArchestrA Reports**.

8. In the **Archestra Reports Database Configuration** window, select **aaFactelligence** for the **Existing SDS Type** (Figure 15 below).


ArchestrA Reports Database Configuration

aaReports Database
Deploy Reports

Deploy Reports - Retrieve or configure shared data source

[✓ Existing Shared Data Source] Create New Shared Data Source Type and Default Data Source

Existing SDS Type: aaFactelligence

New SDS Type:

Default Data Source

Data Source Name: Factelligence

Server Name: localhost

Database Name: Factelligence

Windows Authentication [✓ SQL Server Authentication]

Login Name: sa

Password: ******

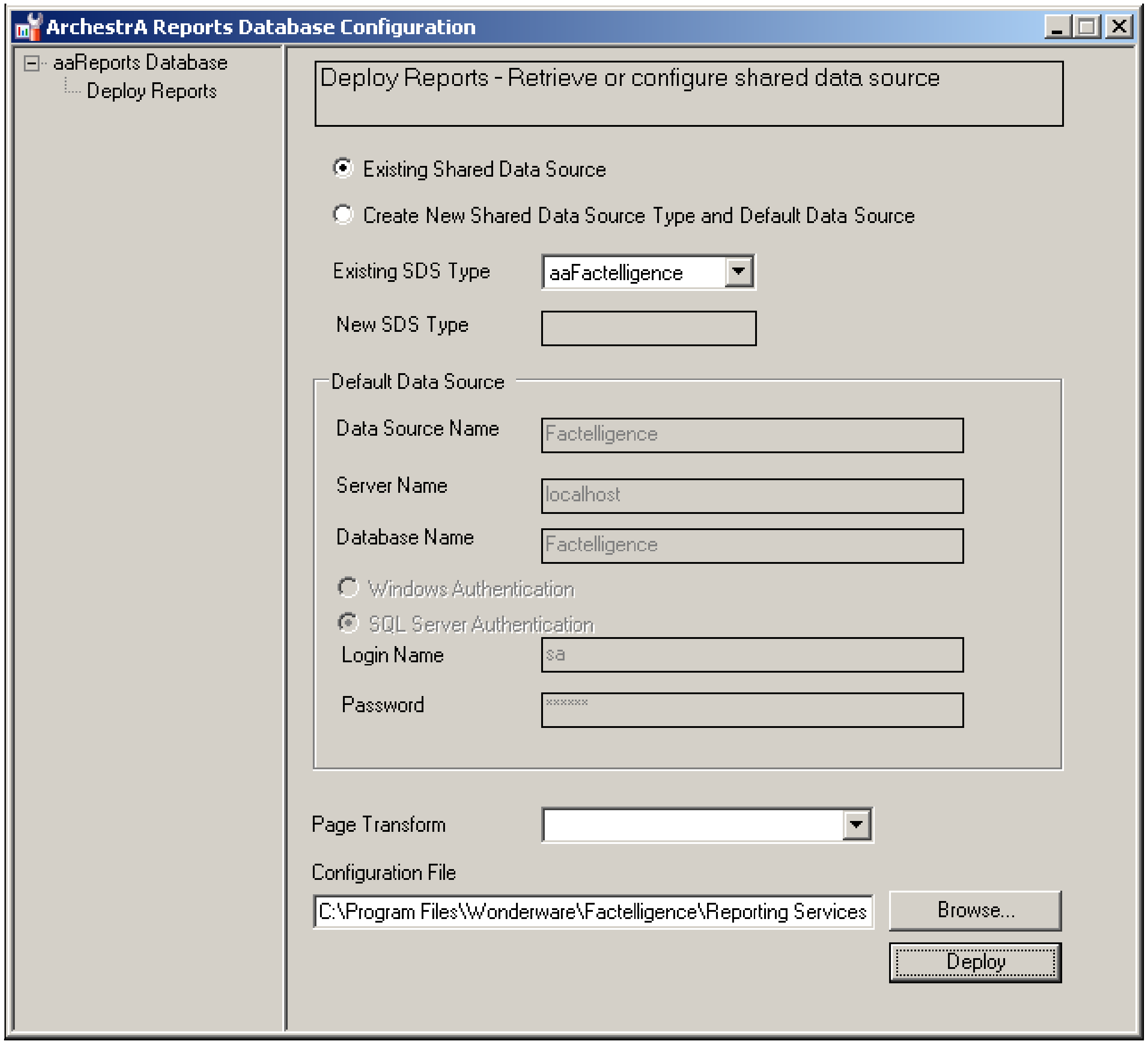
Page Transform:

Configuration File:

Figure 15: Change Existing SDS Type to aaFactelligence

9. Click the **Browse** button to locate the Configuration File.

10. Go to **C:\Program Files\Wonderware\Factelligence\Reporting Services**. The two files that must be deployed are **RptServicesConfig.xml** and **RptServicesConfigEmpty.xml**. The Archestra Reports Database Configuration utility allows only one file at a time to be deployed.


ArchestrA Reports Database Configuration

aaReports Database
Deploy Reports

Deploy Reports - Retrieve or configure shared data source

(● Existing Shared Data Source) ○ Create New Shared Data Source Type and Default Data Source

Existing SDS Type: aaFactelligence

New SDS Type:

Default Data Source

Data Source Name	Factelligence
Server Name	localhost
Database Name	Factelligence

○ Windows Authentication [✓ ● SQL Server Authentication]

Login Name	sa
Password	******

Page Transform:

Configuration File: C:\Program Files\Wonderware\Factelligence\Reporting Services

Figure 16: Deploy Configuration File Path

After both the xml files are deployed, the reports include the new changes.

References

- Tech Note 603 Improving Report Generation Time for Performance 3.5
- Tech Note 616 Performance 3.5 Utilization Capability Object Rollover Counter Behavior and the Changes with Hotfix 1394
- Tech Note 621 Performance 3.5 Utilization Capability Object (UCO) Attribute Extension Considerations

Back to top