



# Plant Applications 8.2

Models



**Proprietary Notice**

The information contained in this publication is believed to be accurate and reliable. However, General Electric Company assumes no responsibilities for any errors, omissions or inaccuracies. Information contained in the publication is subject to change without notice.

No part of this publication may be reproduced in any form, or stored in a database or retrieval system, or transmitted or distributed in any form by any means, electronic, mechanical photocopying, recording or otherwise, without the prior written permission of General Electric Company. Information contained herein is subject to change without notice.

© 2021, General Electric Company. All rights reserved.

**Trademark Notices**

GE, the GE Monogram, and Predix are either registered trademarks or trademarks of General Electric Company.

Microsoft® is a registered trademark of Microsoft Corporation, in the United States and/or other countries.

All other trademarks are the property of their respective owners.

We want to hear from you. If you have any comments, questions, or suggestions about our documentation, send them to the following email address:

[doc@ge.com](mailto:doc@ge.com)

# Table of Contents

|  |    |
|--|----|
| Setting Up the Model Processing Sequence .....                               | 14 |
| Multi-Thread Solution .....  | 14 |
| How It Works .....   | 14 |
| Considerations.....  | 14 |
| Implementation .....   | 14 |
| Model Log Files.....   | 15 |
| Fault Logic.....   | 15 |
| Establishing Fault Logic.....  | 15 |
| Input Tags .....   | 16 |
| Establishing Running Logic.....  | 16 |
| Examples of Running Logic.....   | 17 |
| Building Detection Models with 210, 211, 212 as Templates .....              | 17 |
| Acquidata Models.....  | 19 |
| Interface Objectives .....   | 20 |
| Configuration.....   | 20 |
| Startup and Shutdown .....   | 20 |
| Communication Protocol.....  | 20 |
| Log Files .....  | 21 |
| Model 90-Acquidata Test Data Import.....                                     | 21 |
| Model 90 Properties.....   | 21 |
| Variable Configuration .....   | 22 |
| Test Data Import File Layout .....   | 22 |
| Test Data Import File Layout for Plant Applications-Acquidata Version 1..... | 23 |
| Add Test Transaction Version 1 .....   | 23 |
| Test Data Import File Layout for Plant Applications-Acquidata Version 2..... | 23 |
| Add Test Transaction Version 2 .....   | 23 |
| Test Data Import File Layout for Plant Applications-Acquidata Version 3..... | 24 |
| Add Test Transaction Version 3 .....   | 24 |
| Plant Applications-Acquidata Version 3 File Example.....                     | 25 |
| Test Data Transaction Processing.....  | 26 |
| Test Data Error Recovery .....   | 26 |
| Model 95-Acquidata Sample Export.....  | 27 |

|   |    |
|---|----|
| Model 95 Properties .....                           | 27 |
| Sample Export File Layout.....                      | 28 |
| Sample Export Processing .....                      | 28 |
| Sample Export Error Recovery .....                  | 29 |
| Model 96-Acquidata Grade Specification Export ..... | 29 |
| Model 96 Properties .....                           | 30 |
| Example Output .....                                | 30 |
| Specification Export File Layout .....              | 31 |
| Example Output: .....                               | 31 |
| Specification Export Processing .....               | 32 |
| Specification Export Error Recovery.....            | 32 |
| Autoline Models.....                                | 32 |
| Interface Objectives .....                          | 33 |
| Configuration.....                                  | 33 |
| Startup and Shutdown .....                          | 33 |
| Communication Protocol.....                         | 33 |
| Log Files .....                                     | 33 |
| Model 94-Autoline Sample Export.....                | 34 |
| Model 94 Properties.....                            | 34 |
| Sample Export File Layout.....                      | 34 |
| Version 1 Export Example File Layout.....           | 34 |
| Version 2 Export Example File Layout.....           | 35 |
| Version 3 Export Example File Layout.....           | 35 |
| Version 4 Export Example File Layout.....           | 36 |
| Sample Export Processing .....                      | 36 |
| Model 93-Autoline Sample Export (Table Driven)..... | 37 |
| Model 93 Properties.....                            | 37 |
| Sample Export File Layout.....                      | 37 |
| Version 1 Export Example File Layout.....           | 37 |
| Version 2 Export Example File Layout.....           | 38 |
| Version 3 Export Example File Layout.....           | 38 |
| Version 4 Export Example File Layout.....           | 39 |
| Sample Export Processing .....                      | 39 |

## Models

|  |    |
|--|----|
| Model 89-Autoline Test Data Import.....                                      | 40 |
| Model 89 Properties.....   | 40 |
| Variable Configuration .....   | 40 |
| Test Data Import File Versions 1 to 4 .....                                  | 41 |
| Test Data Import File Layout for Plant Applications-Autoline Version 1 ..... | 41 |
| Test Data Import File Layout for Plant Applications-Autoline Version 2 ..... | 42 |
| Test Data Import File Layout for Plant Applications-Autoline Version 3 ..... | 42 |
| Test Data Import File Layout for Plant Applications-Autoline Version 4 ..... | 43 |
| Sample Test Data Import File for Version 1 .....                             | 44 |
| Sample Test Data Import File for Version 2 .....                             | 45 |
| Sample Test Data Import File for Version 4 .....                             | 48 |
| Configuring Production Unit Cross Reference.....                             | 50 |
| Test Data Transaction Processing.....  | 50 |
| Model 97-Autoline Grade Specification Export.....                            | 50 |
| Model 97 Properties.....   | 51 |
| Grade Spec Export File Layout.....   | 51 |
| Specification Export Processing .....  | 52 |
| Crew Schedule Models .....   | 54 |
| Create a Crew and Shift Schedule from Tags.....                              | 54 |
| Model 1054-Disposition Model.....  | 55 |
| Movement Model 1054 Properties.....  | 56 |
| Model 1054 Stored Procedure Parameters .....                                 | 56 |
| Downtime Models.....   | 57 |
| Downtime Event .....   | 57 |
| Relationships .....  | 57 |
| Downtime Models 200, 210, 211, 212.....                                      | 57 |
| Model Descriptions .....   | 57 |
| Downtime Models Working with Delayed Reasons.....                            | 58 |
| How the Process works .....  | 60 |
| Code for a Downtime Model Delayed Reason .....                               | 60 |
| Model 200.....   | 63 |
| Model 200 Properties.....  | 63 |
| Model 210.....   | 63 |

|  |    |
|--|----|
| Model 211 .....  | 66 |
| Model 212 .....  | 68 |
| Model 211 Location Logic.....  | 71 |
| Establish Location Logic for the Location Determined from Inputs (Model 211) .....                             | 71 |
| Model 212 Location Logic.....  | 71 |
| Establishing Location Logic for the Cause Location Determined By Defining Equipment States<br>(Model 212)..... | 72 |
| Export Models .....  | 72 |
| Model 72 .....   | 73 |
| Model 72 Properties.....   | 73 |
| Model 73 .....   | 73 |
| Model 73 Properties.....   | 73 |
| Model 77 – Data Export.....  | 74 |
| Model 77 Properties.....   | 74 |
| Sample Stored Procedure .....  | 74 |
| Model 78 .....   | 74 |
| Model 78 Properties.....   | 75 |
| Sample Stored Procedure .....  | 75 |
| Model 92 .....   | 76 |
| Model 92 Properties.....   | 76 |
| Model 98 .....   | 76 |
| Model 98 Properties.....   | 76 |
| Model 99 .....   | 77 |
| Model 99 Properties.....   | 77 |
| Genealogy Models 1051-1055.....  | 77 |
| Generic Models 600-603.....  | 77 |
| Generic Model Description and Usage.....   | 77 |
| Generic Model 600-Transition Triggered.....  | 78 |
| Model 600 Description .....  | 78 |
| Model 600 Properties.....  | 78 |
| Model 600 Stored Procedure Parameters .....  | 79 |
| Setting Up a Generic Model 600 .....   | 79 |
| Add a Specific Event Type such as a Production Event Type .....  | 79 |
| Create a Model Template derived from Model 600 .....   | 80 |

## Models

|   |    |
|---|----|
| Assign New Model and the Specific Model Property Values.....    | 80 |
| Setting Up a Generic Model 601 .....                            | 81 |
| Add a Specific Event Type such as a Production Event Type ..... | 81 |
| Create a Model Template derived from Model 601 .....            | 81 |
| Assign New Model and the Specific Model Property Values.....    | 82 |
| Generic Model 601-Archive Value Triggered .....                 | 82 |
| Model 601 Description .....                                     | 82 |
| Model 601 Properties.....                                       | 82 |
| Model 601 Stored Procedure Parameters .....                     | 83 |
| Sample Code .....   | 83 |
| Generic Model 602-Interval Triggered.....                       | 86 |
| Model 602 Description .....                                     | 86 |
| Model 602 Properties.....                                       | 86 |
| Model 602 Stored Procedure Parameters .....                     | 87 |
| Sample Code .....   | 87 |
| Generic Model 603-Multiple Tags, Archive Value Triggered .....  | 89 |
| Model 603 Description .....                                     | 89 |
| Model 603 Properties.....                                       | 90 |
| Model 603 Stored Procedure Parameters .....                     | 90 |
| Model 603 Stored Procedure Header .....                         | 91 |
| Generic Models 604-607 .....                                    | 93 |
| Model 604-ODBC Data Import .....                                | 93 |
| Configuration.....  | 93 |
| Startup and Shutdown .....                                      | 93 |
| Communication Protocol.....                                     | 93 |
| Log Files .....   | 94 |
| Model 604 Description .....                                     | 94 |
| Model 604 Properties.....                                       | 94 |
| Model 605-Generic ODBC Data Export .....                        | 95 |
| Configuration.....  | 95 |
| Startup and Shutdown .....                                      | 95 |
| Communication Protocol.....                                     | 95 |
| Log Files .....   | 95 |

|   |     |
|---|-----|
| Description .....   | 95  |
| Model 605 Properties .....                                  | 95  |
| Sample Stored Procedure .....                               | 97  |
| Model 607-Generic ODBC Data Import.....                     | 99  |
| Configuration.....  | 99  |
| Startup and Shutdown .....                                  | 99  |
| Communication Protocol.....                                 | 99  |
| Log Files .....   | 100 |
| Model 607 Properties.....                                   | 100 |
| Description .....   | 100 |
| Model 650-Event Conformance \ Test Percentage Complete..... | 101 |
| Model 650 Properties.....                                   | 101 |
| Import Models .....   | 101 |
| Model 50 .....  | 101 |
| Model 50 Properties.....                                    | 101 |
| File Layout .....   | 102 |
| Model 51 .....  | 102 |
| Model 51 Properties.....                                    | 102 |
| Model 52 .....  | 102 |
| Model 52 Properties.....                                    | 103 |
| Model 53 .....  | 103 |
| Model 53 Properties.....                                    | 103 |
| Model 54 .....  | 104 |
| Model 54 Properties.....                                    | 104 |
| Model 55 .....  | 105 |
| Model 55 Properties.....                                    | 105 |
| Model 56 .....  | 105 |
| Model 56 Properties.....                                    | 105 |
| Model 57 .....  | 106 |
| Model 57 Properties.....                                    | 106 |
| Model 58 .....  | 106 |
| Model 58 Properties.....                                    | 106 |
| Model 59 .....  | 107 |



## Models

|  |     |
|--|-----|
| Model 59 Properties.....               | 107 |
| Model 64.....                          | 107 |
| Model 64 Properties.....               | 108 |
| Model 65.....                          | 108 |
| Model 65 Properties.....               | 108 |
| Model 66.....                          | 109 |
| Model 66 Properties.....               | 109 |
| Model 67.....                          | 109 |
| Model 67 Properties.....               | 109 |
| Model 68.....                          | 109 |
| Model 68 Properties.....               | 110 |
| Model 69.....                          | 110 |
| Input File Format.....                 | 110 |
| Model 69 Properties.....               | 110 |
| Model 70.....                          | 111 |
| Model 70 Properties.....               | 111 |
| Model 71.....                          | 111 |
| Model 71 Properties.....               | 112 |
| Model 74.....                          | 112 |
| Model 74 Properties.....               | 112 |
| Model 75: Fixed-width File Import..... | 112 |
| Configuration.....                     | 113 |
| Startup and Shutdown.....              | 113 |
| Communication Protocol.....            | 113 |
| Log Files.....                         | 113 |
| Model 75 Description.....              | 113 |
| Model 75 Properties.....               | 114 |
| Model 76.....                          | 114 |
| Model 76 Properties.....               | 114 |
| Model 79 Delimited File Import.....    | 115 |
| Configuration.....                     | 115 |
| Startup and Shutdown.....              | 115 |
| Communication Protocol.....            | 115 |

|  |     |
|--|-----|
| Log Files .....  | 115 |
| Model 79 Description .....                                   | 116 |
| Model Configuration Properties .....                         | 116 |
| Appendix A - Delimited File Import Example .....             | 118 |
| Model 84 .....   | 124 |
| Model 84 Properties .....                                    | 124 |
| Model 85 .....   | 125 |
| Model 85 Properties .....                                    | 125 |
| Model 86 .....   | 125 |
| Model 86 Properties .....                                    | 125 |
| Model 87 .....   | 126 |
| Model 87 Properties .....                                    | 126 |
| Model 91 .....   | 126 |
| Model 91 Properties .....                                    | 126 |
| Configure Model 5013 for Batch Analysis .....                | 127 |
| Using RSBatch.....   | 127 |
| Using Proficy Batch Execution.....                           | 127 |
| Model 1053-Consumption Model .....                           | 129 |
| Consumption Model 1053 Properties .....                      | 129 |
| Input Genealogy Models .....                                 | 130 |
| Model 1052-Input Genealogy Model .....                       | 130 |
| Input Genealogy Model 1052 Properties .....                  | 130 |
| Movement Model 1052 Stored Procedure Parameters .....        | 131 |
| Model 5008 - Input Genealogy Model .....                     | 131 |
| Attach When Running (Based on Time) .....                    | 131 |
| Input Genealogy Model 5008 Properties .....                  | 131 |
| Input Genealogy Model 5008 Stored Procedure Parameters ..... | 131 |
| Model 5009 - Input Genealogy Model .....                     | 131 |
| Match to Event_Num on Output Side .....                      | 131 |
| Input Genealogy Model 5009 Properties .....                  | 132 |
| Input Genealogy Model 5009 Stored Procedure Parameters ..... | 132 |
| Model 5010 - Input Genealogy Model .....                     | 132 |
| Create Output Event Based on Input.....                      | 132 |

## Models

|  |     |
|--|-----|
| Input Genealogy Model 5010 Properties .....                  | 132 |
| Input Genealogy Model 5010 Stored Procedure Parameters ..... | 132 |
| Input Movement Models .....                                  | 132 |
| Model 1051- Input Movement Model .....                       | 133 |
| Movement Model 1051 Properties .....                         | 133 |
| Movement Model 1051 Stored Procedure Parameters .....        | 134 |
| Model 5002- Move Oldest Event to Staged (Historian Tag)..... | 134 |
| Input Movement Model 5002 Properties.....                    | 134 |
| Input Movement Model 5002 Stored Procedure Parameters .....  | 135 |
| Model 5003 - Move Oldest Event to Staged (Variable).....     | 135 |
| Input Movement Model 5002 Properties.....                    | 135 |
| Input Movement Model 5003 Stored Procedure Parameters .....  | 135 |
| Model 5004 - By Event Number (Historian Tag) .....           | 135 |
| Input Movement Model 5004 Properties.....                    | 135 |
| Input Movement Model 5004 Stored Procedure Parameters .....  | 136 |
| Model 5005 - By Event Number (Variable).....                 | 136 |
| By Event_num - Variable .....                                | 136 |
| Input Movement Model 5005 Properties.....                    | 136 |
| Input Movement Model 5005 Stored Procedure Parameters .....  | 136 |
| Model 5006 - By Event Number and Unit.....                   | 136 |
| Input Movement Model 5004 Properties.....                    | 137 |
| Input Movement Model 5006 Stored Procedure Parameters .....  | 137 |
| Model 5007 - By Event Number and Unit (Variables) .....      | 137 |
| Input Movement Model 5005 Properties.....                    | 137 |
| Input Movement Model 5005 Stored Procedure Parameters .....  | 137 |
| Process Order Models .....                                   | 138 |
| Model 1055-Schedule Change Model .....                       | 138 |
| Schedule Change Model 1055 Properties .....                  | 138 |
| Movement Model 1055 Stored Procedure Parameters .....        | 138 |
| Creating Process Orders From Tags .....                      | 138 |
| Product Change Models.....                                   | 139 |
| Model 100 .....  | 139 |
| Model 100 Properties.....                                    | 140 |

|                                       |     |
|---------------------------------------|-----|
| Detecting Product Change Events ..... | 140 |
| Production Event Models .....         | 142 |
| Model 1 .....                         | 142 |
| Model 1 Properties .....              | 142 |
| Model 2 .....                         | 142 |
| Model 2 Properties .....              | 142 |
| Model 3 .....                         | 143 |
| Model 3 Properties .....              | 143 |
| Model 4 .....                         | 144 |
| Model 4 Properties .....              | 144 |
| Model 5 .....                         | 144 |
| Model 5 Properties .....              | 144 |
| Model 6 .....                         | 145 |
| Model 6 Properties .....              | 145 |
| Model 7 .....                         | 146 |
| Model 7 Properties .....              | 146 |
| Model 8 .....                         | 146 |
| Model 8 Properties .....              | 146 |
| Model 9 .....                         | 147 |
| Model 9 Properties .....              | 147 |
| Model 10 .....                        | 148 |
| Model 10 Properties .....             | 148 |
| Model 11 .....                        | 148 |
| Model 11 Properties .....             | 148 |
| Model 12 .....                        | 149 |
| Model 12 Properties .....             | 149 |
| Model 13 .....                        | 150 |
| Model 13 Properties .....             | 150 |
| Model 14 .....                        | 150 |
| Model 14 Properties .....             | 150 |
| Model 15 .....                        | 151 |
| Model 15 Properties .....             | 151 |
| Model 16 .....                        | 151 |

## Models

|                          |     |
|--------------------------|-----|
| Model 16 Properties..... | 152 |
| Model 17.....            | 152 |
| Model 17 Properties..... | 152 |
| Model 18.....            | 153 |
| Model 18 Properties..... | 153 |
| Model 19.....            | 153 |
| Model 19 Properties..... | 153 |
| Model 20.....            | 154 |
| Model 20 Properties..... | 154 |
| Model 21.....            | 155 |
| Model 21 Properties..... | 155 |
| Model 22.....            | 155 |
| Model 22 Properties..... | 155 |
| Model 23.....            | 156 |
| Model 23 Properties..... | 156 |
| Model 24.....            | 156 |
| Model 24 Properties..... | 156 |
| Model 25.....            | 157 |
| Model 25 Properties..... | 157 |
| Model 26.....            | 157 |
| Model 26 Properties..... | 158 |
| Model 27.....            | 158 |
| Model 27 Properties..... | 158 |
| Model 28.....            | 159 |
| Model 28 Properties..... | 159 |
| Model 29.....            | 159 |
| Model 29 Properties..... | 159 |
| Model 30.....            | 161 |
| Model 30 Properties..... | 161 |
| Model 31.....            | 162 |
| Model 31 Properties..... | 163 |
| Model 32.....            | 164 |
| Model 32 Properties..... | 164 |

|                           |     |
|---------------------------|-----|
| Model 33 .....            | 164 |
| Model 33 Properties ..... | 165 |
| Model 34 .....            | 165 |
| Model 34 Properties ..... | 165 |
| Model 35 .....            | 165 |
| Model 35 Properties ..... | 166 |
| Model 36 .....            | 166 |
| Model 36 Properties ..... | 166 |
| Model 37 .....            | 166 |
| Model 37 Properties ..... | 167 |
| Model 38 .....            | 167 |
| Model 38 Properties ..... | 167 |
| Model 39 .....            | 168 |
| Model 39 Properties ..... | 168 |
| Model 40 .....            | 168 |
| Model 40 Properties ..... | 168 |
| Model 41 .....            | 169 |
| Model 41 Properties ..... | 169 |
| Model 42 .....            | 169 |
| Model 42 Properties ..... | 169 |
| Model 43 .....            | 170 |
| Model 43 Properties ..... | 170 |
| Model 44 .....            | 170 |
| Model 44 Properties ..... | 170 |
| Model 45 .....            | 171 |
| Model 45 Properties ..... | 171 |
| Model 46 .....            | 171 |
| Model 46 Properties ..... | 172 |
| Model 47 .....            | 172 |
| Model 47 Properties ..... | 172 |
| Model 48 .....            | 172 |
| Model 48 Properties ..... | 173 |
| Model 49 .....            | 173 |

## Models

|  |     |
|--|-----|
| Model 49 Properties .....                              | 173 |
| Configure Model 118 for Batch Analysis .....           | 174 |
| Model 49000 .....                                      | 175 |
| Detecting Production Events .....                      | 175 |
| Replix Models .....                                    | 177 |
| Model 60 .....   | 177 |
| Model 60 Properties .....                              | 178 |
| Model 61 .....   | 178 |
| Model 61 Properties .....                              | 179 |
| Model 62 .....   | 179 |
| Model 62 Properties .....                              | 179 |
| Model 63 .....   | 179 |
| Model 63 Properties .....                              | 179 |
| Detecting User-Defined Events .....                    | 180 |
| Valmet Models .....                                    | 181 |
| Model 88 .....   | 181 |
| Properties .....                                       | 181 |
| Waste Models .....                                     | 182 |
| Waste Event .....                                      | 182 |
| Relationships .....                                    | 182 |
| Waste Models .....                                     | 183 |
| Log Files .....  | 183 |
| Model 300 – Balancing Waste .....                      | 183 |
| Model 300 Properties .....                             | 183 |
| Model 304 .....  | 184 |
| Model 304 Sample Logic .....                           | 187 |
| Waste Model 304: Waste Occurs on Single Location ..... | 188 |
| Model 5011 - Waste Model .....                         | 191 |
| Waste Model 5011 Properties .....                      | 191 |
| Model 5012 - Waste Model .....                         | 192 |
| Model 5014 - Waste Model .....                         | 192 |

# Setting Up the Model Processing Sequence

In Plant Applications, by default and by design, all models execute in sequence. That is, a model must finish executing before the next model can start. An issue, however, is that a model that may take a few milliseconds to run would be forced to wait for a slower model to finish processing. To help resolve this, a method was needed that would provide the flexibility to allow some models to run sequentially and some models to run in parallel.

## Multi-Thread Solution

To enable some models to run in parallel and other models to run sequentially, the following changes have been made to Plant Applications:

- The Event Manager is multithreaded
- The Database Manager is multithreaded
- The Email Engine is multithreaded
- The Model Processing Group field has been added to models

## How It Works

Initially, the Event Manager uses two threads:

- One thread is used for polling
- One thread is used by model groups with a group number of 0 (zero)

---

**NOTE:** *By default, all models are assigned model group number 0 (zero) and, therefore, all run sequentially.*

---

Each model group gets its own thread and its own database connection. Models within groups are executed sequentially.

By default, multi-threading is turned on. You can turn off multithreading by editing the System User parameters for each of the three services (see the System Users topic) or by editing the Multithreaded site parameter. Note that system user parameters take precedence over site parameters.

## Considerations

It is important to understand that multi-threading does not necessarily mean improved performance. When multiple models are executed at the same time, computer resources are shared. This may cause models running in parallel to run slower. While steps have been taken to prevent deadlocking, you may experience deadlocks, particularly when performing inserts or updates on one of your tables.

If deadlocking is detected by Plant Applications, an attempt will be made to re-run the stored procedure. Additionally, an error message will be written to the Event Manager or Database Manager log files.

Additionally, it is important to remember that each group gets its own thread and its own database connection. Too many threads could have a negative impact on performance.



## Implementation

First, it is recommended that you determine the models that can be run in parallel and the models that must be run in sequence. Then, specify a number (up to six digits) for each group and enter that number in the Model Processing Group field for each model you want to group.



## Models

To enter a Model Processing Group number:

1. In the Plant Applications Administrator, expand Plant Model.
2. Expand the appropriate department and production line.
3. Right-click the production unit and click Configure Events on <production unit>. The Event Detection wizard appears.
4. For more information, please see the Configuring Event Detection topic.
5. Do one of the following:
  - To group an existing model, select the model from the Configured Model list and click  Edit Model.
  - To group a new model, from the Model Type list select the model type that corresponds to the type of event you want to detect and then click Add New Model.
6. On the General tab, type the group number in the Model Processing Group box. The Model Processing Group number can be up to six digits.
7. Finish adding or editing the model and click  to save the configuration.

## Model Log Files

All messages created by models are logged in the Plant Applications Event Manager log files typically in the Proficy\Logfiles directory (this location is a customizable site parameter). The log files are named EventMgr-01.Log and EventMgr-01.Shw with the 01 being the version of the log file. In the "Log" file there will be error messages and in the "Show" file EventMgr.Shw there is a summary of all currently configured models and their current configuration.

## Fault Logic

### Establishing Fault Logic

After determining a location, you must use VBScript to identify the faults for each location. Click the drop-down list of locations (for each location that has faults) and fill in the script. End each script with Fault = <Fault Name>, or Fault = <Fault Value>. If you provide a fault that does not match one already defined in the fault list, the model still logs downtime; however, the fault is ignored.

To search for available faults and automatically fill in either the value or the name, click Insert Fault.... Only those faults already identified for the selected location are displayed.

You can also click **Check Syntax** to check that the script follows proper VBScript syntax.

A fault script can be defined for each location. However, only the fault script for the location determined by location logic will run.

### Fault Logic Example #1

In this example, if alias A = 1 then the fault "Conveyor Fault" is returned; if A = 2 then the fault "Lubrication" is returned.

```
if A = 1 Then Fault = "Conveyor Fault"  
if A = 2 Then Fault = "Lubrication"
```

### Fault Logic Example #2

This example checks to see if the bits are set in a given alias and return the fault accordingly.

```

if A And &H1 Then Fault = 0
if A And &H2 Then Fault = 1
You can also use the following:
if A And &H1 Then Fault = "Conveyor Fault"
if A And &H2 Then Fault = "Lubrication"

```

### Fault Logic Example #3

This example rounds the value in alias A and then uses a case statement to return Fault codes.

```

Select Case Round(A)
Case 0 Fault = "R1 Jam"
Case 1 Fault = "R1 Jam"
Case 2 Fault = "R1 Starve"
Case 4 Fault = "R1 Block"
End Select

```

You can also use the following:

```

Select Case Round(A)
Case 0 Fault = 1
Case 1 Fault = 2
Case 2 Fault = 3
Case 4 Fault = 4
End Select

```

## Input Tags

You can define as many inputs as necessary. For each input, an alias is automatically assigned. When logic is defined for the model, this alias is used to refer to a specific input inside the script (or model logic). The Trigger column determines whether or not a change in value of the specified tag will cause the model to be triggered or evaluated. Items with the trigger off are collected and passed into the model when one of the other tags change. At least one tag must have the trigger on for the model to work. Inputs marked as triggers are constantly monitored by the Event Manager for data changes. Non-trigger inputs are only monitored or read when a trigger tag changes.

Add tags to the model by typing in the tag name from the Tag Search screen. You can press Shift or Ctrl to select multiple tags. You can also use the browse button (...) to the right of the tag to search. You must choose an attribute for each tag. This determines what will be passed in as an input to the model. The only choices are Value and Time-stamp, with Value as the default. You can add the same tag as several different inputs to the same model.

The Sampling Type and Time Offset determine how tag information is collected from the historian and how it will be passed into the model. Choices include Interpolated, Last Good Value, and Next Good Value. In the Precision column, you can set the number of decimal places for the input value of the tag.

You can use the time offset to offset the collection of tag information by plus or minus 'n' seconds relative to the time-stamp that triggered the model to fire (as opposed to the time-stamp the downtime event started, finished, and so forth).

## Establishing Running Logic

## Models

Once you define inputs, these values can be used in VBScripts to determine information about the model, such as the source location. Depending on the logic you are editing, you must include at least one of the statements listed in the following table.

| Model #    | The script determines              | You must include at least one of these statements in the script |
|------------|------------------------------------|---|
| All Models | Whether or not the line is running | Running = True<br>Running = False                               |
| Model 211  | The location                       | Location = "location name"                                      |
| Model 212  | The equipment status               | Status = True<br>Status = False                                 |
| All Models | Faults for each location           | Fault = "Fault name"<br>Fault = Fault value                     |

The running script determines whether or not the production line is running. If the line is not running (Running = False), all subsequent scripts are fired. If the line is running (Running = True), none of the subsequent scripts are fired.

### Examples of Running Logic

#### Running Logic Example #1

In this example of a Running Logic model, **F** is the alias that corresponds to FIX.LINESPEED.F\_CV. This line is said to be running when the LINESPEED tag is greater than 20. The line is *not* running if the LINESPEED is less than 20.

```
if F < 20 then
Running = False
else
Running = True
end if
```

#### Running Logic Example #2

In this example, the line is running when the aliases for **A**, **B**, and **C** are all equal to zero. It assumes, of course, that A, B, and C each have tags corresponding to them (such as FaultTag, ErrorFlagTag, and AlarmTag). If any of the tags have a non-zero value, then the line is assumed to be down; a downtime event is then triggered.

```
if A+B+C THEN
Running = False
else
Running = True
end if
```

## Building Detection Models with 210, 211, 212 as Templates

---

**NOTE:** The functionality in these models is available only if you have purchased the Efficiency Module.

---

If the standard model 200 does not fit your needs, you can customize one of the model templates and provide a unique model name and description. When you customize one of the model templates, you edit the logic that runs that model. The logic for these templates is written in VBScript.

Once you create a new model, that custom model is listed under the Available Models; however, it will not show model properties like the standard model.

Planning for building a new Downtime Detection Model:

- Audit the available signals to trigger downtime information.
- Understand the behavior of each signal as it appears in the historian by charting the historical tags in a chart.
- Map out the logic of the signal data to determine the following:
  - How do I know if the line is running or not?
  - How do I determine the cause location?
  - Given the cause location, how do I determine the individual fault for that location?

To configure a new Downtime model:

1. Click **Plant Model** and browse to the desired production unit.
2. Right-click on the production unit and click **Configure Events on <production unit>**. The **Event Configuration** dialog box appears.
3. Click the **Enable Events** tab.
4. Under **Available Events to Add**, click **Downtime** and then click the **Add Event** button.
5. Click the Configure Event Detection Models tab. Under Events Enabled On This Unit, make sure the Downtime event type is selected.
6. Under **Available Models to Assign**, select one of the following Downtime models:
  - **Model 210:** This model determines Downtime based on a single location only. The location is either running or not running, depending on the VBScript that you enter. To use this model you must have only one location defined.
  - **Model 211:** This model determines downtime by checking various locations to determine if the production line is up or down. It determines the location based on input tag values.
  - **Model 212:** This model determines the cause location by defining the states of each location (or piece of equipment) along the line. Based on how you define the state of each location, the downtime detection model (as opposed to the control system) attempts to determine which piece of equipment along a line was the source or cause of downtime.
7. Click the **Assign Model** button.
8. Under Events Enabled On This Unit, click Downtime.
9. Under **Custom Configurations**, click **New**. The **New Model** dialog box appears.
10. Enter the model name and description. Click **OK**. The **Downtime Model Builder** dialog box appears.

---

*If you plan to copy this model to another production line, do not include the production line name in the model name, because you cannot edit the model name once it is copied. Rather, include a description of the model's logic.*

---

11. On the **Select Model Type** tab, select one of the following model types.
  - **Faults Occur On Single Location (210):** Select this option if you have only one location defined on the Production Line.
  - **Cause Location Determined Directly From Inputs (211):** Select this option to determine the location that caused the Downtime event using the tag data.

- **Cause Location Determined By Defining Equipment States (212):** Select this option to determine the cause location by defining the states of each location (or piece of equipment) along the line.

After determining the model type, you need to define the inputs to the model. These inputs represent the information that you must collect from the control system to determine:

- If the line is down
- The source location
- The fault or root cause

12. Click the **Identify Model Inputs** tab. In the **Tag Search** dialog box, do the following:
  - a. **optional:** Select a different Historian from the **Historian** drop-down list.
  - b. **optional:** In the **Tag Mask** box, type the text you want to use to filter the tag names. For example, if you type **PV**, only the tag names containing **PV** will be returned.
  - c. Click **Search**.
  - d. Select the tag(s) to use as input and click **OK**. For more information, see [Input tags](#).
  - e. Click the **Establish Running Logic** tab and type the VBScript that will define whether the Production Line is running. For more information and to view two examples of running logic, see [Running Logic](#).
13. If you are using Model 211 or Model 212, click the **Establish Location Logic** tab and type the VBScript that will determine the cause location. For more information on location logic and Model 211, see [Model 211 Location Logic](#). For more information on location logic and Model 212, see [Model 212 Location Logic](#).
14. Click the **Establish Fault Logic** tab, do the following:
  - a. Select a unit from the **Location** list.
  - b. Select a fault mode from the **Fault Mode** list.
    - **Assign fault at start change = split:** The fault is assigned immediately when a new downtime record is opened. If a new fault is passed while the initial downtime record is still open, the downtime records are chained together and the assigned fault is passed in as a split.
    - **Assign Fault @ Start:** Assign the fault code when the downtime record is opened.
    - **Assign Fault @ End:** Assign the fault code when the downtime record is closed.
    - **Overwrite fault at End Time:** Assign the fault code at the beginning of the downtime event when opened and if the fault code changes when the record is closed, the downtime record will be updated to reflect the new fault code.
  - c. Write the VBScript that assigns values to each fault. For more information and to view two examples of fault logic, see [Fault Logic](#).

## Acquidata Models

---

**NOTE:** In order to use this model, you must first purchase a license for the Acquidata interface.

---

The Acquidata System automatically captures, analyzes, and tags test data generated by a mill's lab test instruments. The Acquidata transactions supported by Plant Applications are Test Data Import, which sends test result data to, Sample Export, which can be time or event (reel turn-up) based, and Grade Specification Export, which sends test limits to the Acquidata. Data transfers between the two systems are as instantaneous as possible, and occur via FTP.

Plant Applications will be interfaced to the Acquidata System via TCP/IP. The protocol for data transfer will be FTP of fixed format files specific to each type of transaction model supported by the interface.

There will be three transactions supported by the Plant Applications Acquidata Interface. The first will be Test Data Import. Test data will automatically become available real time within the Plant Applications system as tests are exported from Acquidata. The second transaction will be the Sample Export. As Reels become available within Plant Applications, samples will be automatically created for testing within Acquidata. The last transaction supported by the Plant Applications Acquidata interface will be Grade Specification Export.

### **Interface Objectives**

The objective of the interface between Plant Applications and the Acquidata system is to automatically capture data from laboratory instruments typically found in the Dry End Test Lab. By automatically capturing this data, the time burden for testing will be reduced and errors due to manual transposition of data will be eliminated.

Quality testing is a mission critical activity for most mills. For this reason, the Plant Applications Acquidata interface must be extremely reliable, easy to manage and troubleshoot, and perform data transfers in real-time. What this means is that the certain functions of the interface must be completely controlled from within the Proficy Server. Data transfers between the two systems must be as instantaneous as possible.

### **Configuration**

General configuration and specific configuration for each type of transfer for the Plant Applications Acquidata interface will be performed in models and in variables on the Proficy Server using the Plant Applications Administrator.

### **Startup and Shutdown**

The Plant Applications Acquidata interface runs as a component or model inside of the Plant Applications Event Manager service. In order to start or stop the Acquidata Interface, the various models are activated or deactivated in the Event Configuration tables or else through the Plant Applications Administrator Configure Events option.

### **Communication Protocol**

The protocol of communication between the Plant Applications Server and the Acquidata Lab Manager will be structured file transfer using FTP over TCP/IP. The Acquidata Lab Manager computer will have a "Transfer Directory" through which all transactions to and from the Acquidata System will be routed. It can be assumed that both computers will have appropriately configured IP addresses and will have FTP services loaded, tested, and fully functional. The actual transfer of the files can be done using the Plant Applications FTP Engine. This is configured using the Plant Applications Administrator. The testing of sending, retrieval, and deletion of files (Gets) using an FTP client is important prior to installing this interface. Programs are available such as WS\_FTP to test transfers. The Plant Applications FTP Engine is an FTP Client.

Transactions generated by Plant Applications destined for Acquidata will be computer. Once on the Lab Manager computer, the Acquidata system will process files of each transaction type in sorted order by name. Files will be names using the format z9999999.ACQ where z would indicate the transaction type and 9999999 would indicate a unique file number starting at 0000001 and rolling over at 9999999.

It is assumed that transactions generated by Acquidata destined for Plant Applications will remain in the Transfer Directory until purged by the Plant Applications interface. These files will follow the same naming convention as mentioned above.

## Models

The following sections will detail each transaction and the file formats employed. Files will be buffered on the Plant Applications server until successfully transmitted to the File Transfer Directory on the Acquidata System.

### Log Files

All messages logged by the Plant Applications Acquidata interface are logged in the Plant Applications Event Manager log files in the Proficy\Logfiles directory named EventMgr-01.Log with the 01 being the version of the log file. In the "Log" file there will be error messages and in the "Show" file EventMgr.Shw there is a summary of the interface's current configuration. By default, the interface will log only error messages to the Log file including the date and time along with transaction specific data surrounding the error. The Show file will contain details of the current configuration of the interface including which models are activated and their corresponding Master Units (machines) that are configured for Sample Export and also which variables are configured to receive Test Data. The variable configuration will also show the cross-reference to Acquidata Test Codes.

### Model 90-Acquidata Test Data Import

---

**NOTE:** In order to use this model, you must first purchase a license for the Acquidata interface.

---

Test data is created using test instruments and propagated from the Acquidata System to Plant Applications. The AcquidataSystem will construct a Transfer File for each Sample for each Test. Depending on the Test, the File may contain an array of test measurements. Plant Applications can import the average of test measurements, a single array element, or the entire test array. This will be determined by what data is available for an Acquidata Test Code and how the Plant Applications Variable has been configured for that Test Code.

For the purposes of this interface, Test Code and Variable may be considered functionally synonymous, although the Acquidata Work Unit will be appended to the Test Code in order to make it unique across Master Units (machines).

Two types of configuration information will be required to drive Test Data Import transactions. The first type is the model that is specific to receiving test data from Acquidata and the second is the configuration information required for each variable expected to receive data from Acquidata. The model configuration is set up through the Plant Applications Administrator application and the variable configuration information will be stored in the Plant Applications Variables table and also maintained through the Plant Applications Administrator.

### Model 90 Properties

The model configuration required for setting up Test Data Import transactions to receive test data from Acquidata is configured through Model 90. The following model values are required to control the Test Data Import operation of the interface.

| Property                   | Example Value    | Description  |
|----------------------------|------------------|--|
| Maximum Run Time (Seconds) |                  | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group     | Up to six digits | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)    | TINT:1           | Timing Interval Plant Applications checks for New Import Files   |

|                                      |                                |   |
|--------------------------------------|--------------------------------|---|
| Import File Location Spec            | C:\Proficy\Acq\Incoming\X*.Acq | Drive and directory for Import Files                        |
| Accepted File Path                   | C:\Proficy\Acq\Processed       | Drive and directory for Successfully Processed Files        |
| Rejected File Path                   | C:\Proficy\Acq\Unprocessed     | Drive and directory for Rejected Files                      |
| Delete Failed Lookups (Default=TRUE) | FALSE                          | Flag to Delete Files if Test Code Lookup Failed             |
| Purge Days                           | 5                              | # Days of Processed/Unprocessed files to keep               |
| Version (1,2,3)                      | 3                              | Support for Multiple Acquidata Test File Formats (3-Latest) |

### Variable Configuration

Configuration information will be required for each variable expected to receive data from Acquidata. All Variables expected to receive data from Acquidata will be configured with an Acquidata data source. Once Acquidata has been selected as a data source, variable configuration rules within the Administrator will follow those defined for the AutoLog data source (manually entered). The Input\_Tag field will be used to cross-reference Plant Applications variables to Acquidata Test Codes by using the following protocol:

**ACQT:**WorkUnit\TestCode\SampleType/NOSPEC

The constant ACQT: is not required and would indicate that this variable is updated via the Acquidata Test Import Transaction. SampleType would indicate whether the Average, Single Point Average Value, or Array of Average Values is to be retrieved from the Transfer File. The /NOSPEC option is used to disable the specification transfer if it is not desired to pass the specifications for this variable. This is only required if specification transfer has been turned on. The following protocol will be used to indicate the Sample Type

### Variable Sampling Types

**ACQT:**WorkUnit\TestCode\AVG Retrieve Average Of All Readings

**ACQT:**WorkUnit\TestCode\SPV001 Retrieve Position 1 Value

**ACQT:**WorkUnit\TestCode\ARRAY Retrieve Reel Position Average and Array

**ACQT:**WorkUnit\TestCode\AVG/NOSPEC Retrieve Average Of All Readings but do not export the specifications for this variable

Plant Applications-Acquidata Test Data Import using Model 90 has 3 versions. Version 1 and Version 2 use the 10-character Test Code in the variable cross reference to Plant Applications. Version 3 uses the 10-character Test Code plus Qualifier 1-6 containing 6 characters each. This makes the Test Code cross reference lookup a total of up to 46 characters for Version 3.

### Test Data Import File Layout

The following file layout will be used to transmit test data from Acquidata to Plant Applications. Acquidata will place a file for each sample and test into the Transfer Directory on the Acquidata Lab Manager computer. Files will be names using the format z9999999.ACQ where z would be the character indicating a Test Import Transaction (default = X) and 9999999 would indicate a unique file number starting at 0000001 and rolling over at 9999999.

There will be four types of Test Data Import transactions. They are the Add Test, Delete Test, Detail Edit, and Global Edit. The Add Test and Delete Test transactions alter values of single test results. The Detail Edit and Global Edit transactions serve to move data from one sample to



## Models

another. The Detail Edit performs this function on a single test, while the Global Edit transaction performs this function for all tests attached to a sample.

In the case of all transactions, Plant Applications maintains the complete history of test results for each sample. These are the general formats for the Add Test transactions.

### Test Data Import File Layout for Plant Applications-Acquidata Version 1

Plant Applications-Acquidata Test Data Import using Model 90 has 3 versions. Version 1 and Version 2 use the 10-character Test Code in the variable cross reference to Plant Applications. Version 3 uses the 10-character Test Code plus Qualifier 1-6 containing 6 characters each. This makes the Test Code cross reference lookup a total of up to 46 characters for Version 3.

#### Add Test Transaction Version 1

| Field                            | Format                                  | Description                                      |
|----------------------------------|---|--|
| Transaction Type                 | X(1)                                    | A For Add Test Transaction                       |
| Work Unit                        | X(2)                                    | Machine Number\Acquidata Work Unit               |
| Test Code                        | X(10)                                   | Acquidata Test Code                              |
| SID#1                            | X(12)                                   | Grade Code                                       |
| SID#2                            | X(12)                                   | Run Number                                       |
| SID#3                            | X(12)                                   | Reel Number                                      |
| Acquisition Date                 | YYYYMMDD(8)                             | Entered Date                                     |
| Acquisition Time                 | HHMM(4)                                 | Entered Time                                     |
| Average Of Readings              | Z9999.999(10)                           | Average Of Test                                  |
| Number Of Reel Position Averages | Z999(3)                                 | Number Of Reel Position Averages                 |
| Readings                         | Z999(3) +<br>Z9999.999(10) *<br>(n)&&.. | Number Readings In Position AND Position Average |

```
A3 3PHSGGT
.LQ403045      16632      331C1117      200103111419      90.900001001      90.9
00000      0.000000      0.000000      0.000000      0.000000      0.000000
0.000000      0.000000      0.000000      0.000
```

### Test Data Import File Layout for Plant Applications-Acquidata Version 2

Plant Applications-Acquidata Test Data Import using Model 90 has 3 versions. Version 1 and Version 2 use the 10-character Test Code in the variable cross reference to Plant Applications. Version 3 uses the 10-character Test Code plus Qualifier 1-6 containing 6 characters each. This makes the Test Code cross reference lookup a total of up to 46 characters for Version 3.

#### Add Test Transaction Version 2

| Field                 | Format              | Description                                 |
|-----------------------|---------------------|---|
| Transaction Type      | X(1)                | A For Add Test Transaction                  |
| Sidset                | X(1)                | Sample ID Set Number                        |
| Work Unit             | X(2)                | Machine Number\Acquidata Work Unit          |
| SID#1                 | X(12)               | Grade Code                                  |
| SID#2                 | X(12)               | Run Number                                  |
| SID#3                 | X(12)               | Reel Number                                 |
| SID#4                 | X(12)               | Sample ID 4 (Example Tester Name)           |
| Acquisition Date/Time | YYYYMMDD HHMMSS(14) | Entered Date Time                           |
| Test Code             | X(10)               | Acquidata Test Code                         |
| Qualifier 1           | X(6)                | Value of 1 <sup>st</sup> Qualifier Not Used |





added only for troubleshooting purposes. Plant Applications will independently maintain the Grade and Run Number of each sample and does not import these Sample ID values.

Note that all fields are fixed width ASCII text. Text type fields will be right padded with spaces up to the formatted size, numerical fields will be left zero padded up to the formatted size. The Reel Position Averages field is actually an array that repeats the number of readings per position and the position average (n) times up to the Number of Reel Positions.

Each file will contain a single transfer record terminated by a carriage return followed by a line feed.

### **Test Data Transaction Processing**

The Plant Applications Acquidata interface will check for files in the Incoming Directory based on the configured Timing Interval as set in the Import Model 90. The transfer of these files from the Acquidata System to Plant Applications can be achieved using the Plant Applications FTP Engine. The FTP configuration will need to be set up separately using the Plant Applications Administrator to retrieve these files. Files following the format z\*.ACQ where z indicates the Import File Location Spec designated in Model 90 will be processed by the Test Import model. Files will be processed in sequence order by filename. Since a sequence number is attached to each file, files will be sorted alphabetically by filename before processing. Once a file has been processed, the interface will delete it from the Transfer Directory.

The following steps will be completed each processing cycle by the Test Data Import model.

1. Search For Files Names X\*.ACQ and Sort By Name
2. Fetch Files To Local Working Directory
3. Loop Through Each Transaction File
4. Open Transaction File
5. Extract Work Unit and Test Code From File
6. Lookup Variable For Work Unit and Test Code
7. Extract TAPPI Number From File
8. Lookup TAPPI Number in Plant Applications
9. Extract Value(s) From File Based On Variable Configuration
10. Construct and Send Variable Update Message
11. Delete Transaction File Locally and On Acquidata Lab Manager Computer
12. Tests where the acquisition date and time and test values did not change will be ignored.

### **Test Data Error Recovery**

The Plant Applications Acquidata Interface must trap several different error conditions. Each is listed below with the appropriate actions to be taken.

#### **Exception Extracting Fields From File**

**Actions:** Log Error, Copy File To UnProcessed Directory, Delete Transfer File, Move On To Next File

#### **Improper File Format**

**Actions:** Log Error, Copy File To UnProcessed Directory, Delete Transfer File, Move On To Next File

#### **Work Unit and Test Code Not Found**

**Actions:** Log Error, If TestImportKeepNotFound Registry is Set Copy File To UnProcessed Directory, Delete Transfer File, Move On To Next File

## Models

### TAPPI Not Found

**Actions:** Log Error, Copy File To UnProcessed Directory, Delete Transfer File, Move On To Next File

### Error Sending Variable Update

**Actions:** Log Error, Copy File To UnProcessed Directory, Delete Transfer File, Move On To Next File

### Error Deleting Transaction File

**Actions:** Log Error, Copy File To UnProcessed Directory, Attempt To Rename File, If Rename Succeeds, Move On To Next File, Otherwise Shut Down

## Model 95-Acquidata Sample Export

---

**NOTE:** In order to use this model, you must first purchase a license for the Acquidata interface.

---

Events (Reels) will be forwarded to the Acquidata System for defined Master Units (machines) as Events are created in Plant Applications. The Acquidata System will then automatically create samples and make them available for testing. When performing tests, the tester may select which Sample is currently being tested to insure tests are attached to the right Reel.

For the purposes of this interface, "Sample ID" and "Event" may be considered functionally synonymous. Typically, the most import aspect of the Sample Id is the TAPPI or Reel Number, however the Sample ID contains other information about a sample including the Grade, the Run#, and the Turn-up Time of the Reel the sample was derived from.

### Model 95 Properties

The configuration required for setting up Sample Export transactions to transfer reels to Acquidata is configured through Model 95. A Model 95 is required for each Master Production Unit that requires Reel Exports. The following model values are required to control the Sample Export operation of the interface. The model configuration is set up through the Plant Applications Administrator application.

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| First Part Of File                | First Characters in the file   |
| Second Part Of File               |  |
| Construction File Spec            | Construction File Location with First Character Of Sample Export File Names  |
| Outgoing File Path                | File Location after File is Built  |
| (Var Desc) Run Number             | Run Number Variable Description  |
| Last File Sequence #              | Last Sequence # Used in the Filename   |
| Transfer Specs (TRUE)             | TRUE or FALSE. TRUE Turns on Export of Spec Changes each Event   |
| Spec Revision Tag                 | Specification Revision Tag   |
| Spec Level (REJECT,WARNING,ALL)   | Select Specification Limits to be Transferred.   |
| Spec Construction File Spec       | Construction File Location with First Character Of Sample Export File Names  |
| Spec Transfer File Path           | File Location after File is Built  |
| Spec Last File Sequence           | Last Sequence # Used in the Filename   |

|                 |  |
|-----------------|--|
| Run Num Retries |  |
| Version (1,2,3) | Site specific setting <ul style="list-style-type: none"> <li>• 1-Use ProdCode without dashes</li> <li>• 2-Use ProdlId</li> <li>• 3-Use ProdCode</li> </ul> |

### Sample Export File Layout

The following file layout will be used to transmit Sample IDs to Acquidata from Plant Applications. The Plant Applications Acquidata interface will place a file for each sample into the Outgoing Directory on the Plant Applications computer. The transfer of the file from Plant Applications to the Acquidata Lab Manager computer is done using the Plant Applications FTP Engine. Files will be named using the format z9999999.ACQ where z would be the character indicating a Sample Export Transaction (default = S) and 9999999 would indicate a unique file number starting at 0000001 and rolling over at 9999999.

| Field       | Format             | Description  |
|-------------|--------------------|--|
| Promptset   | X(?)               | Character From Model First Part of File              |
| Work Unit   | 9(?)               | Machine Number From Model 2nd Part of File           |
| SID#1       | X(12)              | Run Number   |
| SID#2       | X(12)              | Product Code without dashes, or ProdlId, or ProdCode |
| SID#3       | X(12)              | Reel Number  |
| SID#4       | X(12)              | Blank Padded   |
| Date / Time | YYYYMMDDHHMMSS(14) | Turnup Time  |
| Comment     | X(80)              | Blank Padded   |

11BSTK50

1111

PM1-B1918

20020219141900

Note that all fields are fixed width ASCII text. Text type fields will be right padded with spaces up to the formatted size, numerical fields will be left zero padded up to the formatted size. Each file will contain a single transfer record terminated by a carriage return followed by a line feed.

### Sample Export Processing

The Plant Applications Acquidata interface will be configured to watch for Events on specific Master Units (machines) through the configuration for Model 95. The interface will monitor the "PendingTasks" table on the Proficy Server to determine which events have not yet been processed by the interface. Once an event has been successfully processed, the Plant Applications Acquidata interface will clear its flag in the PendingTasks table and move on to the next event.

For each event the interface determines what must be forwarded to Acquidata, the interface will construct a transfer file according to the format specified in Sample Export File Layout. Files will be named according to the format "z9999999.ACQ" where z indicates the Sample Export Character designated by the Construction File Spec in Model 95. Files are expected to be processed in sequential order. Since a sequence number is attached to each file, files will be sorted alphabetically by filename before processing.

Once a file has been created, the interface will place it in the Outgoing File Path on the Proficy Server. The FTP configuration will need to be set up separately using the Plant Applications Administrator.

Note also that the Sample Export may also trigger a Grade Specification Export based on settings in the model. The Grade Specification Export will occur BEFORE the export of the Sample ID and

## Models

AFTER the Event has been verified to be valid. Details of the Grade Specification export can be found in that section.

The following steps will be completed for each event on the production unit that has Model 95 configured.

1. Detect Event In PendingTasks
2. Determine Master Unit of Event
3. Lookup Timestamp Of Event
4. Lookup Grade Code Of Event Based on Timestamp
5. Lookup Run# of Event Based on Variable Defined in Model for the Timestamp
6. Trigger Grade Specification Export If Turned On
7. Build Sample Export File
8. Place Sample Export File in Outgoing Directory
9. Clear PendingTasks

Since the Acquidata System only processes sample "Adds", changes made in the sample id such as Grade or Run Number are processed as "Adds" by Acquidata. It will be the responsibility of the tester to select the correct sample within Acquidata to attach the test data to.

### Sample Export Error Recovery

The Plant Applications Acquidata Interface must trap several different error conditions surrounding Sample Export. Each is listed below with the appropriate actions to be taken.

#### "Exception Building File"

**Actions:** Log Error, Copy Event File To UnProcessed Directory, Clear Event Config, Move On To Next Event

#### "Event Is <NONE> Grade"

**Actions:** Log Error, Copy Event File To UnProcessed Directory, Clear Event Config, Move On To Next Event

#### "Run# For Event Not Found"

**Actions:** Log Error, Copy Event File To UnProcessed Directory, Clear Event Config, Move On To Next Event

#### "Failure In Grade Spec Export"

**Actions:** Log Error, Bail Out Of Current Event, Don't Clear Event Config, Move On To Next Event

## Model 96-Acquidata Grade Specification Export

---

**NOTE:** In order to use this model, you must first purchase a license for the Acquidata interface.

---

Model 96 is a time-based model and runs at the interval specified by the TINT property. The model creates export files for Acquidata when modifications are made to existing specifications, causing changes to be propagated to the var\_specs table. When the model runs, it captures the value of Last Transfer Time property and runs spserver\_EMGrGet0096Specs passing in the "Last Transfer Time". The results returned by the stored procedure query the var\_specs table capturing any

specification changes between the last transfer time and the current data and time. Model 96 uses the output to create individual specification export files which are then transferred to Acquidata.

**Model 96 Properties**

The following values will be required to control the Specification Export operation of the interface used in Model 96.

| Name                              | Description  |
|-----------------------------------|--|
| Maximum Run Time (Seconds)        | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval to check for Spec Transactions   |
| Spec Revision Tag                 |  |
| Spec Level (REJECT, WARNING, ALL) | Select specification limits to be transferred. <b>REJECT</b> provides Upper and Lower reject limits. <b>WARNING</b> provides Upper and Lower warning limits. <b>ALL</b> provides Upper Reject, Upper Warning, Upper User, Target, Lower User, Lower Warning, Lower Reject limits.                          |
| Construction File Spec            | Construction File Location with First Character Of Sample Export File Names  |
| Transfer File Path                | File Location after File is Built  |
| Transfer Interval (Minutes)       | Timing Interval to build Spec Export Files   |
| Last Transfer Time                | Time of Last Build of Spec Files   |
| Last File Sequence                | Last Sequence # Used in the Filename   |
| Version (1 or 2)                  | Site specific setting<br>1-UseProdCode without dashes<br>2-Use ProId   |

**Example Output**

```
TRQCIMPREV1:ABSTK45          BWT          000050.000T000048.000T
      T000045.000T          T000042.000T000040.000T          F          F
              F          F          F
```

Configuration information will be required for each variable expected to send Specifications to Acquidata. This configuration information will be stored in the Plant Applications Variables table and maintained through the Plant Applications Administrator application.

All Variables expected to send grade specifications to Acquidata will be configured with an Acquidata data source and will have the Work Unit and Test Code embedded in the Input\_Tag field according to the protocol described in the Variable Configuration section. If it is desired to have a variable not send its specifications to Acquidata, then the /NOSPEC parameter needs to be added at the end of the Input\_Tag configuration for that variable.

Note that within Plant Applications, a variable is unique and may have unique specification values defined for each grade. The combination of Work Unit and Test Code are unique within Acquidata, however specifications are only defined to Test Code and Grade level. This will not typically be an issue, as the vast majority of tests performed by Acquidata will not have unique specification values across machines for the same test. If this is a requirement however, different Test Codes will need to be created for the same test on different machines.



## Models

As new Grades are created within Plant Applications, no additional configuration will have to be performed on the Acquadata side to receive QC Limits for the new Grade. The Lab Manager will add the new Grade Codes and Specifications as they are encountered.

### Specification Export File Layout

The following file layout will be used to transmit Grade Specifications to Acquadata from Plant Applications. The Plant Applications Acquadata interface will place a file for each unique Grade Specification (Test Code + Grade) into the Outgoing Directory on the Plant Applications computer. Files will be named using the format z9999999.ACQ where z would be the character indicating a Specification Export Transaction (default = Q) and 9999999 would indicate a unique file number starting at 0000001 and rolling over at 9999999.

| Field           | Format        | Description                                       |
|-----------------|---------------|---|
| Revision Tag    | X(12)         | (Constant) Revision Tag From Model                |
| Operation       | X(1)          | A to Add Limit Record                             |
| Grade Code      | X(12)         | Grade/Product Code                                |
| Test Code       | X(10)         | Test Code From Variables:Input_Tag                |
| Qual1           | X(6)          | Blank   |
| Qual2           | X(6)          | Blank   |
| Upper Reject    | Z9999.999(10) | Upper Reject Limit Setting                        |
| Update URL Flag | X(1)          | T or F - Specifies whether or not to update value |
| Upper Warning   | Z9999.999(10) | Upper Warning Limit Setting                       |
| Update UWL Flag | X(1)          | T or F - Specifies whether or not to update value |
| Upper User      | Z9999.999(10) | Upper User Limit Setting                          |
| Update UUL Flag | X(1)          | T or F - Specifies whether or not to update value |
| Target          | Z9999.999(10) | Target Setting                                    |
| Update TGT Flag | X(1)          | T or F - Specifies whether or not to update value |
| Lower User      | Z9999.999(10) | Lower User Limit Setting                          |
| Update LUL Flag | X(1)          | T or F - Specifies whether or not to update value |
| Lower Warning   | Z9999.999(10) | Lower Warning Limit Setting                       |
| Update LWL Flag | X(1)          | T or F - Specifies whether or not to update value |
| Lower Reject    | Z9999.999(10) | Lower Reject Limit Setting                        |
| Update LRL Flag | X(1)          | T or F - Specifies whether or not to update value |
| Blank           | Z9999.999(10) | Blank Value                                       |
| Flag            | X(1)          | F   |
| Blank           | Z9999.999(10) | Blank Value                                       |
| Flag            | X(1)          | F   |
| Blank           | Z9999.999(10) | Blank Value                                       |
| Flag            | X(1)          | F   |
| Blank           | Z9999.999(10) | Blank Value                                       |
| Flag            | X(1)          | F   |
| Blank           | Z9999.999(10) | Blank Value                                       |
| Flag            | X(1)          | F   |

### Example Output:

```
TRQCIMPREV1:ABSTK45      BWT      000050.000T000048.000T
      T000045.000T      T000042.000T000040.000T      F      F
      F      F      F
```

Note that all fields are fixed width ASCII text. Text type fields will be right padded with spaces up to the formatted size, numerical fields will be left zero padded up to the formatted size.

Each file will contain a single transfer record terminated by a carriage return followed by a line feed.

## Specification Export Processing

The Plant Applications Acquidata interface can be configured to transfer Grade Specifications to the Acquidata either on an event basis or on a time basis.

When transferred on an event basis, ALL specifications that apply to that Event (Reel) will be transferred regardless of when they were last updated. When transferred on a time basis, only those specifications that changed since the last successful processing interval will be propagated to the Acquidata System.

The interface will track the time of the last successful transfer for both event and time triggered transfers. A failure in an event triggered Specification Export will also cause a failure of the Sample ID Export it is associated with. In this instance, the event will not be cleared and the interface will attempt another transfer the next processing cycle.

For each grade specification the interface determines what must be forwarded to Acquidata, the interface will construct a transfer file according to the format specified in the section Specification Export File Layout. Files will be named according to the format z9999999.ACQ where z indicates the character Construction File Spec designated in the Construction File Spec in the model and 9999999 would indicate a unique file number starting at 0000001 and rolling over at 9999999. Files are to be processed in sequential order. Since a sequence number is attached to each file, files will be sorted alphabetically by filename before processing.

Once a file has been created, the interface will place it in the Outgoing Directory on the Proficy Server and can be FTP'd to Acquidata using the Plant Applications FTP Engine. The FTP configuration is set up separately using the Plant Applications Administrator.

The following steps will be completed for each Specification Export.

1. Export Triggered By Event Or Passage of the Spec Export Interval in the model
2. If Export Is Time Based, Determine Specification Changes Since Last Successful Interval For All Acquidata Datasource Variables
3. If Export Is Event Based, Determine Master Unit, Grade, and Timestamp Of Event, Determine Acquidata Datasource Variables on that Unit, and Gather All Specifications Based on Grade and Timestamp of Event.
4. For Each Specification (Variable + Grade), Build Export File Based On Model Setting Spec Export Limits
5. Send Specification Export Files To Outgoing Directory
6. Mark Last Successful Time

## Specification Export Error Recovery

The Plant Applications Acquidata Interface must trap different error conditions surrounding Specification Export. Each is listed below with the appropriate actions to be taken.

### Exception Building File

**Actions:** Log Error, Copy Export File To UnProcessed Directory, Abort Transfer

Succeeds, Move On To Next File, Otherwise Shut Down

## Autoline Models

---

*To use the Autoline models, you must first purchase a license for the Autoline interface.*

---

The Autoline system receives sample and grade specification data from Plant Applications, and returns test data to Plant Applications. It is similar in high level functionality to the AcquiData system.

## Models

There are three transactions supported by the Plant Applications Autoline interface and all three are accomplished through the exchange of formatted data files between Autoline and Plant Applications.

- **Test Data Import:** Test data automatically becomes available within the Plant Applications system as tests are taken. For more information, see the topic, [Model 89-Autoline Test Data Import](#)
- **Sample Export:** As reels, or other testable products become available within Plant Applications, samples will be automatically created for testing within Autoline. For more information, see the topics, [Model 94-Autoline Sample Export](#) and [Model 93-Autoline Sample Export \(Table Driven\)](#).
- **Grade Specification Export:** As samples are created within Autoline, the grade specification limits that apply will be automatically propagated Autoline. For more information, see the topic, [Model 97-Autoline Grade Specification Export](#).

## Interface Objectives

The objective of the interface between Plant Applications and the Autoline system is to automatically capture data from laboratory instruments typically found in the Dry End Test Lab. By automatically capturing this data, the time burden for testing will be reduced and errors due to manual transposition of data will be eliminated.

Quality testing is a mission critical activity for most mills. For this reason, the Plant Applications Autoline interface must be extremely reliable, easy to manage and troubleshoot, and perform data transfers in real-time. What this means is that the functions of the interface must be completely controlled from within the Proficy Server. Data transfers between the two systems must be as instantaneous as possible.

## Configuration

General configuration and specific configuration for each type of transfer for the Plant Applications Autoline interface will be performed in Plant Applications models and on variables in the Proficy Server using the Plant Applications Administrator. (A Plant Applications model performs database manipulations according to configurable model attributes in order to accomplish a set of defined tasks.)

## Startup and Shutdown

The Plant Applications Autoline interface runs as a component or model inside of the Plant Applications Event Manager service. In order to start or stop the Autoline interface, the various models are activated or deactivated in the Event Configuration tables or else through the Plant Applications Administrator Event Configuration Wizard.

## Communication Protocol

The protocol of communication between the Proficy Server and Autoline will be structured file transfer using FTP over TCP/IP. Autoline will have a transfer directory through which all transactions to and from Autoline will be routed. It can be assumed that both the Plant Applications and Autoline computers will have appropriately configured IP addresses and will have FTP services loaded, tested, and fully functional. The actual transfer of the files can be done using the Plant Applications FTP Engine. This is configured using the Plant Applications Administrator. The testing of sending, retrieval, and deletion of files (Gets) using an FTP client is important to verify by using a program such as WS\_FTP prior to commissioning this interface. The Plant Applications FTP Engine is an FTP Client.

## Log Files

All messages logged by the Plant Applications Autoline interface are logged in the Plant Applications Event Manager log files in the Proficy\Logfiles directory named EventMgr-01.Log with the 01 being the version of the log file. In the log file there will be error messages and in the Show file

EventMgr.Shw there is a summary of the interfaces current configuration. By default, the interface will log only error messages to the log file including the date and time along with transaction specific data surrounding the error. The show file will contain details of the current configuration of the interface including which models are activated and their corresponding master units (machines) that are configured for sample export and also which variables are configured to receive test data.

## Model 94-Autoline Sample Export

*In order to use this model, you must first purchase a license for the Autoline interface.*

Events (Reels) will be forwarded to the Autoline system for defined Master Units (machines) as Events are created in Plant Applications. The Autoline system will then automatically create samples and forward the appropriate sample information to the test equipment for testing. When performing tests, the tester may select which Sample is currently being tested from the test equipment to ensure tests are attached to the right Reel.

### Model 94 Properties

The configuration required for setting up Sample Export transactions to transfer reels to Autoline is configured through Model 94. A Model 94 is required for each Master Production Unit that requires Reel Exports. The following model values are required to control the Sample Export operation of the interface. The model configuration is stored in the Plant Applications Event Configuration tables and is configurable through the Plant Applications Administrator application.

| Name                              | Example Value                          | Description  |
|-----------------------------------|--|--|
| <b>Maximum Run Time (Seconds)</b> |  | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Up to six digits                       | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Construction File Spec            | C:\Proficy\Autoline\Construction\?.rel | Construction File Location, extension may vary by site.  |
| Outgoing File Path                | C:\Proficy\Autoline\Outgoing\          | File Location after File is Built  |
| Test Location                     | 1                                      | Test location. For version 4 Test Location, Location Code (41, c)  |
| (Var Desc) Program Number         | PM1 Program Number                     | Program Number Variable Description  |
| Misc                              | 7315                                   | Some code number that Autoline expects   |
| Version (1,2,3,4)                 | 4                                      | There are several different header definitions   |

### Sample Export File Layout

Four versions of sample export file layouts have been defined to date. Example values and field definitions follow. Note that versions 1 and 4 are quite similar, as are versions 2 and 3.

#### Version 1 Export Example File Layout

| Line # | Example Value | Sample Export Version 1 Description          |
|--------|---------------|--|
| 1      | 1             | Paper Machine or sample source, 4 characters |
| 2      | P1-Z0617      | Reel Number, 8 characters                    |

Models

|    |            |   |
|----|------------|---|
| 3  | BSTK50     | Grade definition, must match exactly a Grade name defined in Autoline |
| 4  |            | Leave a blank row   |
| 5  |            | Leave a blank row   |
| 6  |            | Leave a blank row   |
| 7  | 12-06-2001 | Date, 10 characters   |
| 8  | 12:46      | Time, 5 characters  |
| 9  | 1          | Profile program number (very important to get correct)                |
| 10 |            | Leave a blank row   |
| 11 | 7315       | X-scale on graphs, usually PM width in inches?                        |

**Version 2 Export Example File Layout**

| Line # | Example Value    | Sample Export Version 2 Description                |
|--------|------------------|--|
| 1      | [SYSTEM]         | System section name                                |
| 2      | Mode=Insert      | Insert, Update, or Delete                          |
| 3      |                  | Leave a blank row                                  |
| 4      | [Sample]         | Sample section name                                |
| 5      | Date=12-06-2001  | Date, 10 characters                                |
| 6      | Time=12:38       | Time, 5 characters                                 |
| 7      | ProfileProgram=2 | Name of profile program                            |
| 8      | ManualProgram=   | Name of procedure                                  |
| 9      | PM=2             | Name of paper machine                              |
| 10     | Grade=All        | Name of grade                                      |
| 11     | Grammage=BS      | Sample grammage                                    |
| 12     | Field1=Reel      | Field1 Selection Criteria 1, 40 characters - Type  |
| 13     | Field2=2222      | Field2 Selection Criteria 2, 40 characters - Group |
| 14     | Field3=          | Field3 Option 1, 40 characters                     |
| 15     | Field4=          | Field4 Option 2, 40 characters                     |
| 16     | Field5=          | Field5 40 characters                               |
| 17     | Field6=          | Field6 Option 3, 40 characters                     |
| 18     | Field7=P2-Z0616  | Field7 Sample name, 80 characters                  |
| 19     | Field8=          | Field8 80 characters                               |
| 20     | Field9=          | Field9 Position to measure, 80 characters          |
| 21     | Field10=         | Field10 80 characters                              |
| 22     | Field11=         | Field11 80 characters                              |
| 23     | Field12=         | Field12 User name, 80 characters                   |

**Version 3 Export Example File Layout**

| Line # | Example Value   | Sample Export Version 3 Description                |
|--------|-----------------|--|
| 1      | [SYSTEM]        | System section name                                |
| 2      | Mode=Insert     | Insert, Update, or Delete                          |
| 3      |                 | Leave a blank row                                  |
| 4      | [Sample]        | Sample section name                                |
| 5      | Date=12/06/2001 | Date, 10 characters                                |
| 6      | Time=12:47      | Time, 5 characters                                 |
| 7      | ProfileProgram= | Name of profile program                            |
| 8      | ManualProgram=  | Name of procedure                                  |
| 9      | PM=3            | Name of paper machine                              |
| 10     | Grade=BSTK50    | Name of grade                                      |
| 11     | Grammage=All    | Sample grammage                                    |
| 12     | Field1=Bobine   | Field1 Selection Criteria 1, 40 characters - Type  |
| 13     | Field2=3333     | Field2 Selection Criteria 2, 40 characters - Group |

|    |                  |   |
|----|------------------|---|
| 14 | Field3=          | Field3 Option 1, 40 characters            |
| 15 | Field4=          | Field4 Option 2, 40 characters            |
| 16 | Field5=          | Field5 40 characters                      |
| 17 | Field6=          | Field6 Option 3, 40 characters            |
| 18 | Field7=PM3-Z0615 | Field7 Sample name, 80 characters         |
| 19 | Field8=          | Field8 80 characters                      |
| 20 | Field9=          | Field9 Position to measure, 80 characters |
| 21 | Field10=         | Field10 80 characters                     |
| 22 | Field11=         | Field11 80 characters                     |
| 23 | Field12=         | Field12 User name, 80 characters          |

#### Version 4 Export Example File Layout

| Line # | Example Value | Sample Export Version 4 Description                                |
|--------|---------------|--|
| 1      | 4             | Paper Machine or sample source, 4 characters                       |
| 2      |               | Leave a blank row  |
| 3      | BSTK50        | Grade definition, must match exactly a Grade name defined in AL200 |
| 4      | PM4-1M0615    | Reel Number, 8 characters  |
| 5      |               | Leave a blank row  |
| 6      |               | Leave a blank row  |
| 7      | 12-06-2001    | Date, 10 characters  |
| 8      | 12:40         | Time, 5 characters   |
| 9      | 1             | Profile program number (very important to get correct)             |
| 10     |               | Leave a blank row  |
| 11     | 4444          | X-scale on graphs, usually PM width in inches?                     |
| 12     | 4444          | Sample length in mm, integer 00000-16000                           |

#### Sample Export Processing

The Plant Applications Autoline interface will be configured to watch for Events on specific Master Units (machines) through the configuration for Model 94. The interface will monitor the PendingTasks table on the Proficy Server to determine which events have not yet been processed by the interface. Once an event has been successfully processed, the Plant Applications Autoline interface will clear its flag in the PendingTasks table and move on to the next event.

For each event the interface determines what must be forwarded to Autoline, the interface will construct a transfer file according to the format required by Autoline. Files are expected to be processed in sequential order. Since a sequence number is attached to each file, files will be sorted alphabetically by filename before processing.

Once a file has been created, the interface will place it in the Outgoing File Path on the Proficy Server. The FTP configuration will need to be set up separately using the Plant Applications Administrator.

The following steps will be completed for each event detected by the Sample Export Model 94.

1. Detect Event In PendingTasks
2. Determine Master Unit of Event
3. Lookup Timestamp Of Event
4. Lookup Grade Code Of Event Based on Timestamp
5. Lookup Program # of Event Based on Variable Defined in Model for the Timestamp
6. Build Sample Export File

## Models

7. Place Sample Export File in Outgoing Directory
8. Clear PendingTasks

Customized specification and event exports can be created using Result Set Type 50 to create any needed export file definition, if model 94 or model 97 described below is not sufficient.

### Model 93-Autoline Sample Export (Table Driven)

*In order to use this model, you must first purchase a license for the Autoline interface.*

Events (Reels) will be forwarded to the Autoline system for defined Master Units (machines) as Events are found in Plant Applications at the end of the timing interval. The Autoline system will then automatically create samples and forward the appropriate sample information to the test equipment for testing. When performing tests, the tester may select which Sample is currently being tested from the test equipment to insure tests are attached to the right Reel.

#### Model 93 Properties

The configuration required for setting up Sample Export transactions to transfer reels to Autoline is configured through Model 93. A Model 93 is required for each Master Production Unit that requires Reel Exports. The following model values are required to control the Sample Export operation of the interface. The model configuration is stored in the Plant Applications Event Configuration tables and is configurable through the Plant Applications Administrator application.

| Name                              | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks the table for new files to export.  |
| Construction File Spec            | Construction File Location, extension may vary by site.  |
| Outgoing File Path                | File Location after File is Built  |
| Test Location                     | Test location. For version 4 Test Location, Location Code (41, c)  |
| (Var Desc) Program Number         | Program Number Variable Description  |
| Misc                              | Some code number that Autoline expects   |
| Version (1,2,3,4)                 | There are several different header definitions   |

#### Sample Export File Layout

Four versions of sample export file layouts have been defined to date. Example values and field definitions follow. Note that versions 1 and 4 are quite similar, as are versions 2 and 3.

##### Version 1 Export Example File Layout

| Line # | Example Value | Sample Export Version 1 Description                                   |
|--------|---------------|---|
| 1      | 1             | Paper Machine or sample source, 4 characters                          |
| 2      | P1-Z0617      | Reel Number, 8 characters   |
| 3      | BSTK50        | Grade definition, must match exactly a Grade name defined in Autoline |
| 4      |               | Leave a blank row   |
| 5      |               | Leave a blank row   |
| 6      |               | Leave a blank row   |

|    |            |  |
|----|------------|--|
| 7  | 12-06-2001 | Date, 10 characters                                    |
| 8  | 12:46      | Time, 5 characters                                     |
| 9  | 1          | Profile program number (very important to get correct) |
| 10 |            | Leave a blank row                                      |
| 11 | 7315       | X-scale on graphs, usually PM width in inches?         |

### Version 2 Export Example File Layout

| Line # | Example Value    | Sample Export Version 2 Description                |
|--------|------------------|--|
| 1      | [SYSTEM]         | System section name                                |
| 2      | Mode=Insert      | Insert, Update, or Delete                          |
| 3      |                  | Leave a blank row                                  |
| 4      | [Sample]         | Sample section name                                |
| 5      | Date=12-06-2001  | Date, 10 characters                                |
| 6      | Time=12:38       | Time, 5 characters                                 |
| 7      | ProfileProgram=2 | Name of profile program                            |
| 8      | ManualProgram=   | Name of procedure                                  |
| 9      | PM=2             | Name of paper machine                              |
| 10     | Grade=All        | Name of grade                                      |
| 11     | Grammage=BS      | Sample grammage                                    |
| 12     | Field1=Reel      | Field1 Selection Criteria 1, 40 characters - Type  |
| 13     | Field2=2222      | Field2 Selection Criteria 2, 40 characters - Group |
| 14     | Field3=          | Field3 Option 1, 40 characters                     |
| 15     | Field4=          | Field4 Option 2, 40 characters                     |
| 16     | Field5=          | Field5 40 characters                               |
| 17     | Field6=          | Field6 Option 3, 40 characters                     |
| 18     | Field7=P2-Z0616  | Field7 Sample name, 80 characters                  |
| 19     | Field8=          | Field8 80 characters                               |
| 20     | Field9=          | Field9 Position to measure, 80 characters          |
| 21     | Field10=         | Field10 80 characters                              |
| 22     | Field11=         | Field11 80 characters                              |
| 23     | Field12=         | Field12 User name, 80 characters                   |

### Version 3 Export Example File Layout

| Line # | Example Value    | Sample Export Version 3 Description                |
|--------|------------------|--|
| 1      | [SYSTEM]         | System section name                                |
| 2      | Mode=Insert      | Insert, Update, or Delete                          |
| 3      |                  | Leave a blank row                                  |
| 4      | [Sample]         | Sample section name                                |
| 5      | Date=12/06/2001  | Date, 10 characters                                |
| 6      | Time=12:47       | Time, 5 characters                                 |
| 7      | ProfileProgram=  | Name of profile program                            |
| 8      | ManualProgram=   | Name of procedure                                  |
| 9      | PM=3             | Name of paper machine                              |
| 10     | Grade=BSTK50     | Name of grade                                      |
| 11     | Grammage=All     | Sample grammage                                    |
| 12     | Field1=Bobine    | Field1 Selection Criteria 1, 40 characters - Type  |
| 13     | Field2=3333      | Field2 Selection Criteria 2, 40 characters - Group |
| 14     | Field3=          | Field3 Option 1, 40 characters                     |
| 15     | Field4=          | Field4 Option 2, 40 characters                     |
| 16     | Field5=          | Field5 40 characters                               |
| 17     | Field6=          | Field6 Option 3, 40 characters                     |
| 18     | Field7=PM3-Z0615 | Field7 Sample name, 80 characters                  |



## Models

|    |          |   |
|----|----------|---|
| 19 | Field8=  | Field8 80 characters                      |
| 20 | Field9=  | Field9 Position to measure, 80 characters |
| 21 | Field10= | Field10 80 characters                     |
| 22 | Field11= | Field11 80 characters                     |
| 23 | Field12= | Field12 User name, 80 characters          |

### Version 4 Export Example File Layout

| Line # | Example Value | Sample Export Version 4 Description                                |
|--------|---------------|--|
| 1      | 4             | Paper Machine or sample source, 4 characters                       |
| 2      |               | Leave a blank row  |
| 3      | BSTK50        | Grade definition, must match exactly a Grade name defined in AL200 |
| 4      | PM4-1M0615    | Reel Number, 8 characters  |
| 5      |               | Leave a blank row  |
| 6      |               | Leave a blank row  |
| 7      | 12-06-2001    | Date, 10 characters  |
| 8      | 12:40         | Time, 5 characters   |
| 9      | 1             | Profile program number (very important to get correct)             |
| 10     |               | Leave a blank row  |
| 11     | 4444          | X-scale on graphs, usually PM width in inches?                     |
| 12     | 4444          | Sample length in mm, integer 00000-16000                           |

### Sample Export Processing

The Plant Applications Autoline interface will be configured to watch for Events on specific Master Units (machines) through the configuration for Model 94. The interface will monitor the PendingTasks table on the Proficy Server to determine which events have not yet been processed by the interface. Once an event has been successfully processed, the Plant Applications Autoline interface will clear its flag in the PendingTasks table and move on to the next event.

For each event the interface determines what must be forwarded to Autoline, the interface will construct a transfer file according to the format required by Autoline. Files are expected to be processed in sequential order. Since a sequence number is attached to each file, files will be sorted alphabetically by filename before processing.

Once a file has been created, the interface will place it in the Outgoing File Path on the Proficy Server. The FTP configuration will need to be set up separately using the Plant Applications Administrator.

The following steps will be completed for each event detected by the Sample Export Model 93.

1. Detect Event In PendingTasks
2. Determine Master Unit of Event
3. Lookup Timestamp Of Event
4. Lookup Grade Code Of Event Based on Timestamp
5. Lookup Program # of Event Based on Variable Defined in Model for the Timestamp
6. Build Sample Export File
7. Place Sample Export File in Outgoing Directory
8. Clear PendingTasks

Customized specification and event exports can be created using Result Set Type 50 to create any needed export file definition, if model 93, 94 or 97 described is not sufficient. **Appendix A** has been added to show developers how to export specifications and events to an AL300 by calling a stored procedure and use result set 50 to create the export file rather than using model 97.

## Model 89-Autoline Test Data Import

*In order to use this model, you must first purchase a license for the Autoline interface.*

Test data will be created by Autoline, which will construct a Transfer File for each Sample for each Test. Depending on the Test, the File may contain an array of test measurements. Plant Applications may be concerned with simply the average of test measurements, a single array element, or the entire test array. This will be determined by what data is available for an Autoline Test Code and how the Plant Applications Variable has been configured for that Test Code.

Two types of configuration information will be required to drive Test Data Import transactions. The first type is the model that is specific to receiving test data from Autoline and the second is the configuration information required for each variable expected to receive data from Autoline. The model configuration is stored in the Plant Applications Event Configuration tables and is configurable through the Plant Applications Administrator application and the variable configuration information will be stored in the Plant Applications Variables table and also maintained through the Plant Applications Administrator.

### Model 89 Properties

The model configuration required for setting up Test Data Import transactions to receive test data from Autoline is configured through Model 89. The following model values are required to control the Test Data Import operation of the interface.

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Min)               | Timing Interval Plant Applications checks for New Import Files   |
| Import File Location              | Drive and directory, extensions may vary   |
| Accepted File Path                | Drive and directory for Successfully Processed Files   |
| Rejected File Path                | Drive and directory for Rejected Files   |
| Purge Days                        | # Days of Processed/Unprocessed files to keep  |
| Version (1,2,3, 4)                | Support for Multiple Autoline Test File Formats (4-Latest)   |

### Variable Configuration

Configuration information will be required for each variable expected to receive data from Autoline. All Variables expected to receive data from Autoline will be configured with an Autoline data source. Once Autoline has been selected as a data source, variable configuration rules within the Administrator will follow those defined for the Autolog data source (manually entered). The Input Tag field will be used to cross-reference Plant Applications variables to Autoline Test Codes by using the following protocol:

#### Variable Sampling Types for Versions 1 and 4

Test Location\PropertyNumber\S1 Retrieve Average Of All Readings

Test Location\PropertyNumber\S2 Retrieve Standard Deviation of all readings

## Models

Test Location\PropertyNumber\S3 Retrieve Minimum Of All Readings

Test Location\PropertyNumber\S4 Retrieve Maximum Of All Readings

Test Location\PropertyNumber\v1 Retrieve Individual Test Input Tag

Example for versions 1 and 4: 1\1\S1

### Variable Sampling Types for Versions 2 and 3

Test Location\PropertyNumber\AVE Retrieve Average of values

Test Location\PropertyNumber\NUM Retrieve Number of values

Test Location\PropertyNumber\MIN Retrieve Minimum value

Test Location\PropertyNumber\MAX Retrieve Maximum value

Test Location\PropertyNumber\VAL1 Retrieve Individual test value

Example for versions 2 and 3: PM2\500\AVE

### Test Data Import File Versions 1 to 4

Use the following tables to match your test data import file format to one of the supported Plant Applications versions.

#### Test Data Import File Layout for Plant Applications-Autoline Version 1

For the Input Tag Configuration use the format [Test Location\Property#\S1].

| Line# | Example   | Description   | Needed by Plant Applications |
|-------|-----------|---|------------------------------|
| 1     | 41        | Paper Machine/Sample Source (4 characters) Table Prod_Units needs a unit with an Extended_Info of PM x And a 2nd unit with the Extended_Info of SET x | Yes                          |
| 2     | 19J1526   | Reel Number, 8 characters   | Yes                          |
| 3     | INF70     | Grade definition  | No                           |
| 4     |           | Grammage, 8 characters  | No                           |
| 5     |           | Order, 12 characters  | No                           |
| 6     |           | Run, 12 characters  | No                           |
| 7     | 9/15/1999 | Date, 10 characters, Turnup Date  | No                           |
| 8     | 14:46     | Time, 5 characters, Turnup Time   | No                           |
| 9     | 1         | Measurement schedule number   | No                           |
| 10    | 2         | Profile program number  | No                           |
| 11    | 7300      | X-scale on graphs, usually PM width in inches   | No                           |
| 12    | 7315      | Sample length in mm, integer 00000-16000  | No                           |
| 13    | 0         | Left trim, integer 00000-16000  | No                           |
| 14    | 0         | Right trim, integer 00000-16000   | No                           |
| 15    |           |   | No                           |
| 16    |           |   | No                           |
| 17    |           | Number of Properties  | Yes                          |
| 18    |           | Property #  | Yes                          |
| 19    |           | Ave,Std,Min,Max,#Appr,#Readings,Error   | Yes                          |
| 20    |           | Targets/Limits  | Yes                          |
| 21    |           | #Values/Actual Data Points  | Yes                          |
| 22    |           | 22&& #18-21 repeated # of times indicated in record #17   | Yes                          |

**Test Data Import File Layout for Plant Applications-Autoline Version 2**

For the Input Tag Configuration use the format [Test Location\Property#\AVE].

| <b>Line#</b> | <b>Example</b> | <b>Description</b>   | <b>Needed by Plant Applications</b> |
|--------------|----------------|--|-------------------------------------|
| 1            | AL300          | AL300 must be somewhere in Line 1  | Yes                                 |
| 2            | PM2            | Machine Number(The Extended_Info in Table Prod_Units must match this value, this is used in searching for the Reel#) | Yes                                 |
| 3            |                |  | No                                  |
| 4            |                |  | No                                  |
| 5            |                |  | No                                  |
| 6            |                |  | No                                  |
| 7            |                |  | No                                  |
| 8            |                |  | No                                  |
| 9            |                |  | No                                  |
| 10           | 2001-12-01     | Date, 10 characters, Turnup Date   | No                                  |
| 11           | 15:17:00       | Time, 5 characters, Turnup Time  | No                                  |
| 12           |                |  | No                                  |
| 13           |                |  | No                                  |
| 14           |                |  | No                                  |
| 15           |                |  | No                                  |
| 16           |                |  | No                                  |
| 17           |                |  | No                                  |
| 18           | PM2-Z0613      | Reel #   | Yes                                 |
| 19           |                |  | No                                  |
| 20           |                |  | No                                  |
| 21           |                |  | No                                  |
| 22           |                |  | No                                  |
| 23           |                |  | No                                  |
| 24           |                |  | No                                  |
| 25           | 670            | Sample Number?   | Yes                                 |
| 26           |                |  | No                                  |
| 27-41        | PROP-SUMMARY   | Property Summaries   | Yes                                 |
| 42-60        | PROP-VALUES    | Property Values  | Yes                                 |
| 61-          | PROP-SUMMARY   | Paired Summary-Value sections repeat as needed   | Yes                                 |

**Test Data Import File Layout for Plant Applications-Autoline Version 3**

For the Input Tag Configuration use format [Test Location\Property#\AVE].

| <b>Line#</b> | <b>Example</b> | <b>Description</b>   | <b>Needed by Plant Applications</b> |
|--------------|----------------|--|-------------------------------------|
| 1            | AL300          | AL300 must be somewhere in Line 1  | Yes                                 |
| 2            | 1              | Machine Number(The Extended_Info in Table Prod_Units must match this value, this is used in searching for the Reel#) | Yes                                 |
| 3            |                |  | No                                  |
| 4            |                |  | No                                  |

## Models

|       |              |  |     |
|-------|--------------|--|-----|
| 5     |              |  | No  |
| 6     |              |  | No  |
| 7     |              |  | No  |
| 8     |              |  | No  |
| 9     |              |  | No  |
| 10    | 2            | Date, 10 characters, Turnup Date               | No  |
| 11    | 7300         | Time, 5 characters, Turnup Time                | No  |
| 12    |              |  | No  |
| 13    |              |  | No  |
| 14    |              |  | No  |
| 15    |              |  | No  |
| 16    |              |  | No  |
| 17    |              |  | No  |
| 18    |              | Reel #   | Yes |
| 19    |              |  | No  |
| 20    |              |  | No  |
| 21    |              |  | No  |
| 22    |              |  | No  |
| 23    |              |  | No  |
| 24    |              |  | No  |
| 25    |              |  | No  |
| 26    |              |  | No  |
| 27-41 | PROP-SUMMARY | Property Summaries                             | Yes |
| 42-46 | PROP-VALUES  | Property Values                                | Yes |
| 47-   | PROP-SUMMARY | Paired Summary-Value sections repeat as needed | Yes |

### Test Data Import File Layout for Plant Applications-Autoline Version 4

For the Input Tag Configuration use format [Test Location\Property#S1].

| Line# | Example   | Description   | Needed by Plant Applications |
|-------|-----------|---|------------------------------|
| 1     | 41        | Paper Machine or sample source (4 characters)   | Yes                          |
| 2     | 3721      | Unknown (12 characters)   | No                           |
| 3     | INF70     | Grade definition, must match exactly a Grade name defined in Autoline   | No                           |
| 4     | 19J1526   | Reel Number, 8 characters   | Yes                          |
| 5     |           |   | No                           |
| 6     | D         | Test Location(12 characters) (Table Prod_units requires Extended_Info = 'MACH=D' to use in searching for the Reel#) | Yes                          |
| 7     | 9/15/1999 | Date, 10 characters, Turnup Date  | No                           |
| 8     | 14:46     | Time, 5 characters, Turnup Time   | No                           |
| 9     | 1         | Measurement schedule number   | No                           |
| 10    | 2         | Profile program number  | No                           |
| 11    | 7300      | X-scale on graphs, usually PM width in inches   | No                           |
| 12    | 7315      | Sample length in mm, integer 00000-16000  | No                           |
| 13    | 0         | Left trim, integer 00000-16000  | No                           |
| 14    | 0         | Right trim, integer 00000-16000   | No                           |

|    |     |   |     |
|----|-----|---|-----|
| 15 | JIM | Optional field displayed on screen, this is entered by the operator from the keyboard | No  |
| 16 |     | Flag for All Data being in File(True=1)   | Yes |
| 17 |     | Number of Parameters  | Yes |
| 18 |     | Property #  | Yes |
| 19 |     | Ave,Std,Min,Max,#Appr,#Readings,Error(Tab delimited)                                  | Yes |
| 20 |     | Targets/Limits  | No  |
| 21 |     | #Values/Individual Values(Tab delimited)  | Yes |
| 22 |     | 22&& #18-21 repeated # of times indicated in record #17                               | Yes |

**Sample Test Data Import File for Version 1**

|           |  |  |  |  |  |  |
|-----------|--|--|--|--|--|--|
| 1         |  |  |  |  |  |  |
| P1-Z0608  |  |  |  |  |  |  |
| BSTK55    |  |  |  |  |  |  |
| 51        |  |  |  |  |  |  |
|           |  |  |  |  |  |  |
|           |  |  |  |  |  |  |
| 12/5/2001 |  |  |  |  |  |  |
| 9:07      |  |  |  |  |  |  |
| 1         |  |  |  |  |  |  |
| 1         |  |  |  |  |  |  |
| 350       |  |  |  |  |  |  |
| 8890      |  |  |  |  |  |  |
| 0         |  |  |  |  |  |  |
| 40        |  |  |  |  |  |  |
| STICK     |  |  |  |  |  |  |
| 1         |  |  |  |  |  |  |
| 18        |  |  |  |  |  |  |

Models

|       |       |       |       |       |   |   |
|-------|-------|-------|-------|-------|---|---|
| 1     |       |       |       |       |   |   |
| 84.56 | 0.207 | 84.3  | 84.8  | 4     | 4 | 0 |
| 82.5  | 80.5  | 100   | 50    | 100   |   |   |
| 4     | 84.3  | 84.61 | 84.8  | 84.53 |   |   |
| 5     |       |       |       |       |   |   |
| 90.63 | 0.088 | 90.52 | 90.73 | 4     | 4 | 0 |
| 50    | 0     | 100   | 0     | 100   |   |   |
| 4     | 90.52 | 90.66 | 90.73 | 90.61 |   |   |
| 6     |       |       |       |       |   |   |
| 1.12  | 0.095 | 1.01  | 1.24  | 4     | 4 | 0 |
| 10    | -10   | 10    | -10   | 10    |   |   |
| 4     | 1.24  | 1.1   | 1.01  | 1.13  |   |   |

**Sample Test Data Import File for Version 2**

|  |
|--|
| AL300 SAMPLE 2001-12-01 15:56:17, LW080262.AL1 |
| PM2  |
| 40-2500-07-670                                 |
| 530  |
| 40   |
| 40   |
| 0  |
| 12/1/2001                                      |
| 15:24:00                                       |

|              |
|--------------|
| 12/1/2001    |
| 15:17:00     |
| Reel         |
| 2001         |
| NA           |
| NA           |
| NA           |
| NA           |
| PM2-Z0613    |
| NA           |
| NA           |
| NA           |
| NA           |
| NA           |
| 670          |
| NA           |
| 25           |
| PROP-SUMMARY |
| 200          |
| 200          |
| Porosity     |
| NA           |



## Models

|             |      |
|-------------|------|
|             | 1    |
|             | 165  |
|             | 19   |
|             | 14   |
|             | 137  |
|             | 193  |
|             | 0    |
|             | 1    |
|             | 1    |
|             | 14   |
| PROP-VALUES |      |
|             | 480  |
|             | 193  |
|             | 800  |
|             | 167  |
|             | 1120 |
|             | 191  |
|             | 1440 |
|             | 176  |
|             | 1760 |
|             | 169  |
|             | 2080 |

|      |
|------|
| 176  |
| 2400 |
| 180  |
| 2720 |
| 156  |
| 3040 |
| 184  |
| 3360 |
| 137  |
| 3680 |
| 142  |
| 4000 |
| 138  |
| 4320 |
| 153  |
| 4640 |
| 147  |

**Sample Test Data Import File for Version 4**

|         |  |  |  |  |  |  |  |  |  |
|---------|--|--|--|--|--|--|--|--|--|
| 41      |  |  |  |  |  |  |  |  |  |
| 3766    |  |  |  |  |  |  |  |  |  |
| DSTK55  |  |  |  |  |  |  |  |  |  |
| 11E2312 |  |  |  |  |  |  |  |  |  |

Models

|           |      |      |      |      |      |      |      |        |               |
|-----------|------|------|------|------|------|------|------|--------|---------------|
|           |      |      |      |      |      |      |      |        |               |
| C         |      |      |      |      |      |      |      |        |               |
| 5/23/2001 |      |      |      |      |      |      |      |        |               |
| 22:39     |      |      |      |      |      |      |      |        |               |
| 1         |      |      |      |      |      |      |      |        |               |
| 2         |      |      |      |      |      |      |      |        |               |
| 7264      |      |      |      |      |      |      |      |        |               |
| 7315      |      |      |      |      |      |      |      |        |               |
| 0         |      |      |      |      |      |      |      |        |               |
| 0         |      |      |      |      |      |      |      |        |               |
| JAMAL     |      |      |      |      |      |      |      |        |               |
| 1         |      |      |      |      |      |      |      |        |               |
| 3         |      |      |      |      |      |      |      |        |               |
| 1         |      |      |      |      |      |      |      |        |               |
| 25.4      | 0.57 | 24.6 | 26.1 | 11   | 11   | 0    |      |        |               |
| 0         | 0    | 0    | 0    | 0    |      |      |      |        |               |
| 11        | 25.7 | 24.9 | 24.6 | 24.9 | 26.1 | 25.8 | 26   | 25..   | 3 more values |
| 2         |      |      |      |      |      |      |      |        |               |
| 25.3      | 0.4  | 24.8 | 25.9 | 11   | 11   | 0    |      |        |               |
| 0         | 0    | 0    | 0    | 0    |      |      |      |        |               |
| 11        | 25.6 | 25.2 | 25.1 | 25.2 | 24.8 | 25.7 | 24.8 | 24.9.. | 3 more values |
| 3         |      |      |      |      |      |      |      |        |               |

|      |       |      |      |      |      |     |      |      |                  |
|------|-------|------|------|------|------|-----|------|------|------------------|
| 2.44 | 0.045 | 2.34 | 2.5  | 21   | 21   | 0   |      |      |                  |
| 0    | 0     | 0    | 0    | 0    |      |     |      |      |                  |
| 21   | 2.34  | 2.47 | 2.38 | 2.44 | 2.47 | 2.5 | 2.47 | 2.48 | ..13 more values |

### Configuring Production Unit Cross Reference

In order for Plant Applications to send incoming test information to the appropriate production unit for tests run on paper from different areas, (where test variables have been configured) the prod\_units table must be updated during the configuration processes. The format of this update query is Update prod\_units SET extended\_info = x WHERE pu\_id = y. Use SELECT \* from prod\_units to determine the pu\_id of the production unit that should receive the test data. Refer to the 4 layouts, lines 1, 2, 2, and 6 respectively for the extended info values. Note that in layout 1, test data can be sent to two units, so extended\_info needs to be updated on two production units. The second unit, called the SET unit is not optional.

Examples:

Update Prod\_units Set Extended\_Info='MACH=R' where pu\_id=96

Update Prod\_units Set Extended\_Info='MACH=S1' where pu\_id=39

### Test Data Transaction Processing

The Plant Applications Autoline interface will check for files in the Incoming Directory based on the configured Timing Interval as set in the Import Model 89. The transfer of these files from the Autoline computer to the Plant Applications computer can be achieved using the Plant Applications FTP Engine. The FTP configuration will need to be set up separately using the Plant Applications Administrator to retrieve these files.

The following steps will be completed each processing cycle by the Test Data Import model.

1. FTP engine delivers files to working directory named in model 89 configuration
2. Files are selected based on filename mask in model configuration
3. Loop Through Each Transaction File
4. Open Transaction File
5. Extract Machine and Property Number From File
6. Extract TAPPI Number From File
7. Lookup TAPPI Number in Plant Applications
8. Lookup Variable For Machine and Property Number
9. Extract Value(s) From File Based On Variable Configuration
10. Construct and Send Variable Update Message
11. Delete Transaction File Locally and On AcquiData Lab Manager Computer

### Model 97-Autoline Grade Specification Export

*In order to use this model, you must first purchase a license for the Autoline interface.*

The Plant Applications Autoline interface may be configured to export grade specifications to the Autoline system on a time basis. The grade specifications that are sent from Plant Applications to

## Models

Autoline are dependent on which variables in Plant Applications are configured under the Autoline data source. Limits that are supported are reject limits and entry limits.

Customized specification and event exports can be created using Result Set 50 to create any needed export file definition, if model 97 is not sufficient. For more information, see the topic, Result Set 50.

### Model 97 Properties

The configuration required for setting up specification export transactions to transfer specifications to Autoline is configured through Model 97. The model configuration is stored in the Plant Applications Event Configuration tables and is configurable through the Plant Applications Administrator. The following values will be required to control the specification Export operation of the interface used in Model 97.

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval to check for Spec Transactions   |
| Construction File Spec            | Construction File Location with First Character Of Sample Export File Names  |
| Outgoing File Path                | File Location after File is Built  |
| Transfer Interval (Minutes)       | Timing Interval to build Spec Export Files   |
| Last Transfer Time                | Time of Last Build of Spec Files   |

All variables expected to send grade specifications to Autoline will be configured with an Autoline data source.

---

**NOTE:** Model 97 looks for variables that have the input tag containing the string "v1". These variables are configured with model 89, versions 1 and 4 only. Model 97 will also export specifications only for variables that have the input tag containing the string "v1." This means that only the individual test variable specifications will be exported.

---

### Grade Spec Export File Layout

The grade specification export file layout definition for each grade is as follows:

| Values | Description  |
|--------|--------------|
| BSTK55 | Product Code |
| 1      | Always a 1   |
| 1      | Property Id  |
| 1      | Always a 1   |
| 1      | Always a 1   |
| 32     | Target       |
| 0      | Lower Reject |
| 100    | Upper Reject |
|        | Lower Entry  |
|        | Upper Entry  |
| 10     | Always a 10  |
| 12     | Always a 12  |

|   |   |
|---|---|
| 1 | Now it's start over at the next Property Id again |
|---|---|

### Specification Export Processing

The interface will track the time of the last successful transfer for time triggered transfers.

For each grade specification, the interface determines what must be forwarded to Autoline, the interface will construct a transfer file.

Once a file has been created, the interface will place it in the outgoing directory on the Plant Applications system and can be FTP'ed to the Autoline system using the Plant Applications FTP engine. The FTP configuration will need to be set up separately using the Plant Applications Administrator.

The following steps will be completed for each specification export.

1. Check for specification transactions based on the timing interval. If there has been a transaction, then determine if any Autoline variables have been impacted.
2. For each product/grade, build an export file based on the model settings and the specification export limits for each Autoline variable that has specifications defined for that grade.
3. Send specification export files to the outgoing directory after the transfer interval setting has passed.
4. Mark the Last Successful Time in the parameters for Model 97 to be used to make sure the transfer interval has passed before creating export files again.

The following table contains a sample grade specification layout.

| Line # | Example Value      | Sample Spec Export Description   |
|--------|--------------------|--|
| 1      | [SYSTEM]           | System section name  |
| 2      | Mode=Insert        | Insert, Update, or Delete  |
| 3      | [Grade]            | Grade section name   |
| 4      | GradeName          | Grade Name   |
| 5      | GradeNumber        | Grade Number   |
| 6      | PMName             | Paper machine name   |
| 7      | Grammage           | Sample grammage  |
| 8      | ProfileProgram     | Name of profile program  |
| 9      | ManualProgram      | Name of manual program   |
| 10     | ReadOnly           | Yes to allow grade to be edited, No to disallow edits  |
| 11     | Option1            | Option1 text   |
| 12     | Option2            | Option2 text   |
| 13     | Option3            | Option3 integer  |
| 14     | Option4            | Option4 integer  |
| 15     | Option5            | Option5  |
| 16     | NumberOfProperties | Number of defined properties to follow   |
| 17     | [GRADEPROP]        | Grade Property section name  |
| 18     | PROP0001           | Comma delimited list of Property, set value, level 1 low, level 1 high, level 2 low, level 2 high, level 3 low, level 3 high, and report (1=print, 0=no print) |
| 19     | PROP0002           | Same   |
| 20     | PROP0003           | Same   |
| 21     | PROP0004           | Same   |
| 22     | PROP0005           | Same   |
| 23     | PROPnnnn           | Same   |

## Models

The following table contains a sample grade specification file, which Plant Applications would export to Autoline in order to create a new grade within Autoline. Note that the only parameter under the SYSTEM section can be set to Update or Delete as well as New.

```
[SYSTEM]

Mode=New

[GRADE]

GradeName=30-2500-00-070

GradeNumber=30# MANDO 70 MF BL WHITE

PMName=PM5

Grammage=30

ProfileProgram=070

ManualProgram=

ReadOnly

Option1=

Option2=

Option3=0

Option4=0

Option5=0

NumberOfProperties=10

[GRADEPROP]

PROP0001=1300,30.00,29.00,31.00,27.90,32.10,25.80,34.20,1

PROP0002=633,70.00,68.50,71.50,67.90,72.10,65.80,74.20,1

PROP0003=654,0.00,-1.00,1.00,-1.00,2.00,-1.00,4.00,1

PROP0004=641,88.00,86.00,100.00,85.50,100.00,80.50,100.00,1
```

```

PROP0005=300,3.80,3.50,4.40,3.10,4.70,2.40,5.60,1
PROP0006=301,3.80,3.50,4.40,3.10,4.70,2.40,5.60,1
PROP0007=1200,30.00,24.00,62.00,22.00,62.00,14.00,62.00,1
PROP0008=800,11.50,10.50,40.00,9.20,40.00,6.00,40.00,1
PROP0009=623,0.3192,0.3142,0.3242,0.3142,0.3242,0.3142,0.3242,1

```

## Crew Schedule Models

Model 801 creates crew and shift records from tags. When the shift tag value changes, Model 801 retrieves the new value from the shift tag and crew tag and checks for an existing record for the current time. If no record exists, then a new record is created. If any records exist between the start time and end time of the new record, the existing records are deleted and replaced with the new record.


If gaps are allowed between shifts (the Chain Shifts option is selected), then the start time is the time of the tag change and the end time is the start time plus the value of the shift time parameters.

---

**NOTE:** *There is no crew message for the Message Bus.*

---

### Create a Crew and Shift Schedule from Tags

15. In the  Plant Model, right-click the production unit where you want to create a crew and shift schedule and click **Detect Events on <production unit>**. The **Event Detection** wizard appears.
16. On the **Configured Models** tab, select **Crew Schedule** from the **Model Type** list and click **Add New Model**. Three new tabbed pages appear: **General**, **Identify Input(s)**, and **Scripts**.
17. On the **General** tab, do the following:
  - a. **optional:** In the **Maximum Run Time (Seconds)** box, type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop).
  - b. **optional:** Type the group number in the Model Processing Group box. The Model Processing Group number can be up to six digits.
  - c. **optional:** In the **Extended Information** box, type additional information that may be useful. This is not used on any reports or displays.
  - d. **optional:** In the **Exclusions** box, type the values this model is to ignore and not use to trigger the model. Use a comma to separate values.
  - e. If gaps in the shift are allowed, select **Chain Shift Times**.
18. Click the **Identify Input(s)** tab to select tags as inputs.
  - a. If the tag is to be set as a trigger, select the **Trigger** check box. A minimum of one trigger tag is required.






- b. Click the **Browse** button beside the **Tag** box. The **Tag Search** dialog box appears. Click **Search** (you can enter search criteria prior to clicking the button if necessary). Select the tag and click **OK**.
- c. From the **Attribute** list, select either Value or Timestamp, depending on whether you want the value or time of the tag to be passed.
- d. From the **Sampling Type** list, select the type of sampling that is applied to the tag. For more information on sampling types, see Sampling Types.
- e. In the **Time Offset** box, enter the number of seconds backward from the trigger time that the next value will be retrieved.

For example:

If you have trigger tag A in the list and you also have another tag B that is used as an input in the VB Script, you can specify what the time offset will be for tag B. This would be the amount of time backward from the trigger time that the EventMgr would attempt to get the 'Last Good Value' for tag B. So if tag A changed at 9/28/07 7:00, and you specified a 10 second offset for tag B, then the EventMgr would attempt to retrieve the Last Good Value for tag B starting at 9/28/07 6:59:50.

19. Click the **Scripts** tab to edit sample scripts or write script logic. On each tab, a sample script is provided, which you can use, or you can click to write script logic.

The following functionality is available on each tab.

- Click . The **Script Builder** dialog box appears where you can edit the existing script or to write script logic.
- In the **Script Builder** dialog box:
- Click . The **Select Alias** dialog box appears. Select the Alias and click **OK**. The Alias letter appears in the **Script** box.
- Click  **Check Syntax** to ensure you have entered the VB Script correctly.
- Click the **Define Crew Logic** tab to edit the script that generates the crew name when the event is triggered. It will support the expression "Crew=NULL" which will signify that there is no record in the Crew/Shift table and will prevent the model from firing.
- Click the **Define Shift Logic** tab to edit the script that generates the shift name when the event is triggered. It will support the expression "Shift=NULL" which will signify that there is no record in the Crew/Shift table.
- Click the **Define Duration Logic** tab to edit the script that generates duration when the event is triggered. It will support the expression "Duration=0" which will signify that there is no record in the Crew/Shift table. The default is **8**.

7. Click  to activate the model.

## Model 1054-Disposition Model

Disposition models are used to update the event status of events and are triggered:

- When the value of a defined historian tag(s) (TriggerTag) changes.
- When the value of a defined variable(s) (Trigger Var Id) changes.
- When an event is created or the event status is changed.

---

*The functionality in this model is available only if you have purchased the Production Management Module.*

---

### Movement Model 1054 Properties

The following model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TriggerTag(s)                     | The tag or tags used to trigger the model. A value change on any of the defined tags will trigger the model and the stored procedure is called. In the Plant Applications Administrator, a value from this tag is passed into the stored procedure based on the chosen sampling type.                      |
| Trigger Var Id(s)                 | The variables or variables used to trigger the model. A value change on any of the defined variables will trigger the model and the stored procedure is called and the variable value is passed into the stored procedure.   |
| Local SP Name                     | Stored procedure called when the model is triggered.   |

### Model 1054 Stored Procedure Parameters

The following parameters are required at the beginning of the stored procedure that is called by Model 1054. The parameter variables can be named whatever is desired but the order must be maintained.

| Variable Name                       | Parameter Description  |
|-------------------------------------|--|
| @ReturnStatus int OUTPUT,           | Flag to indicate Success or Failure (1-Success,0-Failure)  |
| @ReturnMessage varchar(255) OUTPUT, | Error Message to Write to Log File   |
| @EC_ID int,                         | Input to the Stored Procedure indicating the EC_Id of the model from the Event_Configuration table.  |
| @JumpToTime Datetime OUTPUT,        | This is used to allow you to control how far back the EventManager goes in time to look for tag changes after a reload or restart of the Event Manager. By default the Event Manager goes back 3 days and looks for tags changes. In your code you could find the latest event on your unit and set the JumpToTime to be a second after this and then have it start looking from there for new events. |
| @ReservedInput1 varchar(30),        | Reserved   |
| @ReservedInput2 varchar(30),        | Reserved   |
| @ReservedInput3 varchar(30),        | Reserved   |
| @ReservedInput4 varchar(30),        | Reserved   |
| @TriggerTag varchar(10),            | The Alias of the Tag which Triggered the SP  |
| @TriggerTag_OldValue varchar(25),   | The Previous Value of the Triggering Tag   |
| @TriggerTag_OldTime Datetime,       | The Previous Time of the Triggering Tag  |
| @TriggerTag_NewValue varchar(25),   | The New Value of the Triggering Tag  |
| @TriggerTag_NewTime Datetime,       | The New Time of the Triggering Tag   |
| @Hist1Alias varchar(30),            | The Alias Letter For Tag or Variable 1   |
| @Hist1Value varchar(30),            | The Value For Tag or Variable 1  |
| @Hist2Alias varchar(30),            | The Alias Name For Tag or Variable 2   |
| @Hist2Value varchar(30)             | The Value For Tag or Variable 2  |

|                          |  |
|--------------------------|--|
| .....continued as needed |  |
|--------------------------|--|

## Downtime Models

### Downtime Event

A Downtime event occurs either when equipment is not running (Downtime) or when equipment is not running at its target rate (rate loss). When tracking Downtime, the key measure is Downtime minutes. When tracking rate loss both Downtime minutes and the lost opportunity from target production rate are the key measures. A Downtime event represents the time a particular unit (or line) was in a faulted condition.

You must have the license for the Efficiency Management module to configure downtime event detection.

### Relationships

**Unit:** The major piece of equipment around which Downtime is being tracked.

**Fault:** What the control system thought the reason for Downtime was. Can be a "First Out," or simply the active interlock that prevents equipment from running.

**Detail:** An individual Downtime event whose duration represents the time of a given fault condition. An overall Downtime occurrence may have several faults; therefore, several details.

**Summary:** An overall Downtime event which contains one or more Downtime details found. A summary represents the total time of Downtime, whereas the detail represents the time in a given fault condition.

**Location:** The specific piece of equipment (equipment module) along a production line causing the line to go down.

**Cause Reasons:** The reasons thought to be the cause of a Downtime event.

**Action Reason:** Reasons identifying any corrective action taken.

### Downtime Models 200, 210, 211, 212

The downtime detection models are the logic that determines when the line is running and when downtime is occurring. Detection models process input signals to determine if the production line is down, the source location of downtime, and the downtime fault. The fault can then be used to default the cause reasons for a particular downtime event.

### Model Descriptions

There are 5 Downtime models (2 standard models and 3 model templates). You cannot edit the standard model 200, but you can edit the logic in the model templates for Models 210, 211, 212.

| Downtime Model Type                                 | Description  |
|---|--|
| Produces Delay When Not (Running) Model (Model 200) | This is a standard Downtime model. It determines downtime based on the input tag that you select. While you cannot edit this model's logic, you can set its properties. Since you can only edit this model's properties, it does not appear as a model type in the Downtime Model Builder. |
|   |  |

|   |   |
|---|---|
| Faults Occur On Single Location (Model 210)*                        | This model determines downtime based on a single location only. The location is either running or not running, depending on the VBScript that you enter. To use this model you must have only one location defined. When you have only one location defined, this model is the only option enabled on the Downtime Model Builder              |
| Cause Location Determined Directly From Inputs (Model 211)*         | This model determines downtime by checking various locations to determine if the production line is up or down. It determines the location based on input tag values.   |
| Cause Location Determined by Defining Equipment States (Model 212)* | This model determines the cause location by defining the states of each location (or piece of equipment) along the line. Based on how you define the state of each location, the downtime detection model (as opposed to the control system) attempts to determine which piece of equipment along a line was the source or cause of downtime. |

\* **Maximum # of tags:** 500

**Maximum size of script:** 7000 characters (no restriction on Administrator)

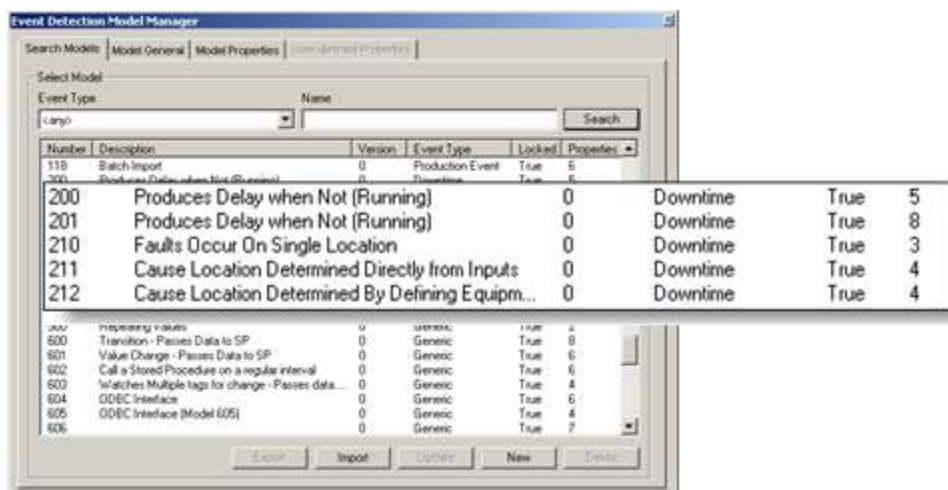
## Downtime Models Working with Delayed Reasons

Standard operation for downtime models requires a fault code to be supplied in real time at the time the fault is detected. At some sites the *control system* may require the reason a machine stopped before it can resume operation. However, for several standalone machines, e.g. CNC, the machine may stop without transmitting a reason or fault to Plant Applications.

Following are an overview procedure and code that enable the process to provide the reason later through Historian (instead of through complex custom screens). This streamlines the process for many manufacturing operations, especially where QuickPanels are an advantage, e.g. CNC and injection molding operations.

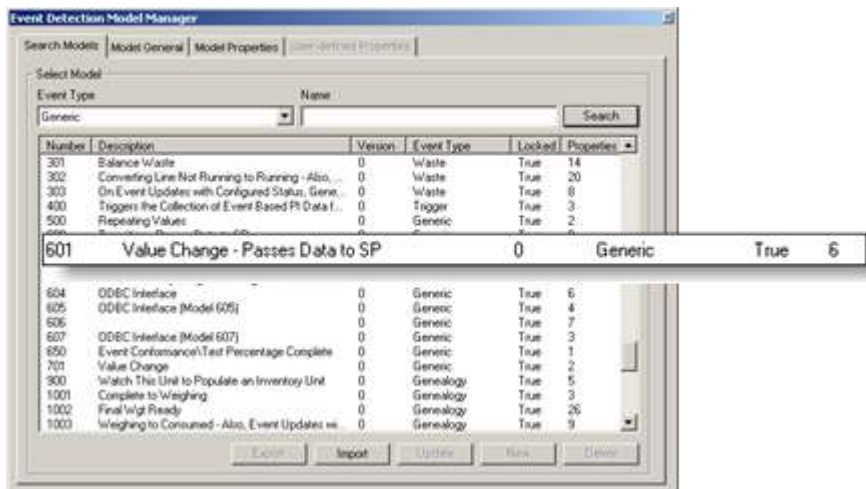
Procedure to delay the reason for a downtime model:

1. Select any existing Plant Applications Downtime model

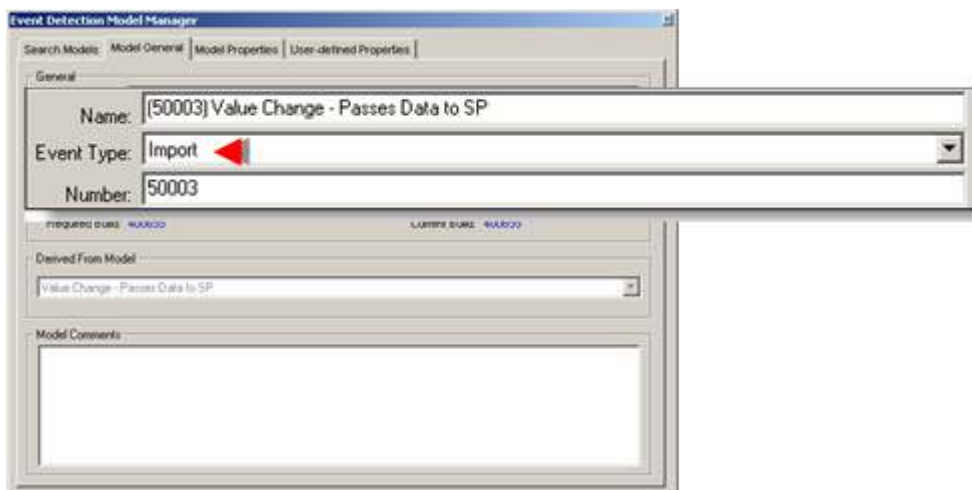


2. Create a Model 50XXX Import type based on Model 601 (*Delayed Downtime Fault Entry Model*).
  - a. Select Model 601.
  - b. Click **New**.

## Models

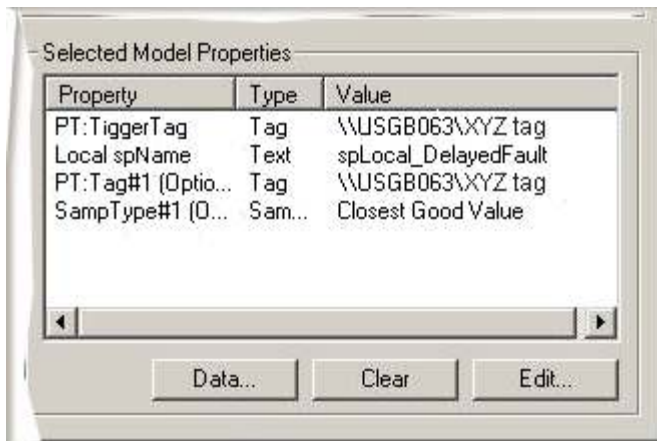


- c. Select the Model General tab.
- d. Select **Import** in the **Event type** field for the new 50XXX model.



3. Include the following in the Plant Applications Downtime model configuration.
  - Map a Historian **real time** tag twice that can detect the time the machine stops and starts. In this process, the Downtime model determines what time the machine stopped.
  - Configure the translation table.
 

The Downtime model will not use the translation table for this process; however the Import type model, which cannot have its own translation table, will use it.
4. Make sure the following four parameters are configured for the model.



- a. Trigger Tag.
- b. Stored procedure for the model to use.
- c. Fault value tag (usually the same as the trigger tag is step a).
- d. Sampling type for retrieving the value from the fault value tag (step c).

*Review **Selected Model Properties** on the **Configure Event Detection Models** tab in a production unit's **Event Configuration** dialog box.*

**How the Process works**

**NOTE:** Proficy uses the code to capture the reason through the real time Historian tag and translates it via the Fault Translation Table configured for the unit. Proficy assumes that the last Downtime event is the event this reason explains.

| Delay Reason Process   | Example  |
|--|--|
| 1. The Downtime model real time tag value changes, indicating that the machine has stopped.                              | RT tag value changes from <b>0</b> to <b>1</b> . |
| 2. The Downtime model detects the time.  | <b>3:02P</b>                                     |
| 3. The Model 50XXX Fault tag receives a value.   | <b>4</b>   |
| 4. The Model 50XXX code:   |  |
| a. Uses the Translation table to translate the Fault tag value into a reason. (The Fault tag time stamp is disregarded.) | <b>Maintenance</b>                               |
| b. Finds the most recent downtime event.   | <b>3:02P</b>                                     |
| c. Assumes that the latest Fault value provides the reason for the downtime event.                                       | <b>Maintenance</b>                               |

**Code for a Downtime Model Delayed Reason**

```

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
/***** Object: Stored Procedure dbo.spLocal_DelayedFault      Script Date:
7/19/2004 9:35:21 AM *****/

```

## Models

```
if exists (select * from dbo.sysobjects where id =
object_id(N'[dbo].[spLocal_DelayedFault]') and OBJECTPROPERTY(id, N'IsProcedure') =
1)
drop procedure [dbo].[spLocal_DelayedFault]
GO
CREATE PROCEDURE dbo.spLocal_DelayedFault
@ReturnStatus int OUTPUT,
@ReturnMessage varchar(255) OUTPUT,
@ECId int,
@HistorianTimeStamp datetime,
@HistorianTagValue int
AS
Declare
@DetailId Int,
@PUID int,
@FaultID int,
@CurrentFaultID int,
@EndTime datetime,
@CurrentFaultValue varchar(25),
@SPUID int,
@R1 int,
@R2 int,
@R3 int,
@R4 int
--find the pu_id from the ec_id of the model calling this sp
Select @PUID = null
Select @PUID = PU_Id From Event_Configuration Where EC_Id = @ECId
If @PUID is null
    goto error
--find the latest TED record for this PU_ID
Select @CurrentFaultID = null
Select top 1 @DetailId = TEDet_Id,@EndTime = End_Time,@CurrentFaultID = TEFault_ID
from timed_event_Details
where @puid = pu_id and Start_Time < @HistorianTimeStamp
order by Start_Time desc
--ifTED already has a fault dont change it
If @CurrentFaultID is not null
    Begin
    Select @CurrentFaultValue = null
    Select @CurrentFaultValue = TEFault_Value
    from Timed_Event_Fault
    where PU_ID = @PUID and @CurrentFaultID = TEFault_ID
    goto NoProcessing
    End
--Make sure the hist tag fault code value exists in the fault trans table for this
model and pu
Select @FaultID = null
Select
@SPUID = Source_PU_ID,
@FaultID = TEFault_ID,
@R1 = Reason_Level1,
@R2 = Reason_Level2,
@R3 = Reason_Level3,
@R4 = Reason_Level4
```

```

from Timed_Event_Fault
where PU_ID = @PUID and TEFault_Value = convert(varchar(25),@HistorianTagValue)
If @FaultID is null
    goto NoFault
--if there is not already a fault for this latest downtime event, then use the one
from the histtag
If @CurrentFaultID is null
-- update timed_event_details set TEFault_ID = @FaultID where TEDet_ID = @DetailId
Begin
    -- Send Out An UPDATE Message To Set Fault Of Current Record
    Select ResultsetType = 5,
        PU_Id = @PUID,
        Source_PU_Id = @SPUID,
        StatusId = d.TEStatus_Id,
        FaultId = @FaultId,
        Reason1 = @R1,
        Reason2 = @R2,
        Reason3 = @R3,
        Reason4 = @R4,
        Prod_Rate = null,
        Duration = d.Duration,
        TransType = 2, -- update
        StartTime = d.Start_Time,
        EndTime = d.End_Time,
        TEDet_Id = @DetailId
    From Timed_Event_Details d
    Where TEDet_Id = @DetailId
End
Done:
Select @ReturnMessage = 'Delayed Fault ' + convert(varchar(5),@HistorianTagValue) +
' at ' + convert(varchar(25),@HistorianTimeStamp) + ' successfu for TEDETID ' +
convert(varchar(25),@DetailID)
Select @ReturnStatus = 1
Return
NoProcessing:
Select @ReturnMessage = 'Delayed Fault ' + convert(varchar(5),@HistorianTagValue) +
' at ' + convert(varchar(25),@HistorianTimeStamp) + ' but pre existing fault ' +
convert(varchar(25),@CurrentFaultValue)
Select @ReturnStatus = 0
Return
NoFault:
Select @ReturnMessage = 'Delayed Fault ' + convert(varchar(5),@HistorianTagValue) +
' at ' + convert(varchar(25),@HistorianTimeStamp) + ' is not a valid fault'
Select @ReturnStatus = 0
Return
Error:
Select @ReturnMessage = 'Delayed Fault ' + convert(varchar(5),@HistorianTagValue) +
' at ' + convert(varchar(25),@HistorianTimeStamp) + ' but some Error'
Select @ReturnStatus = 0
Return
GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

```



## Models

```
GRANT EXECUTE ON [dbo].[spLocal_DelayedFault] TO [public]
GO
GRANT EXECUTE ON [dbo].[spLocal_DelayedFault] TO [comxclient]
GO
GRANT EXECUTE ON [dbo].[spLocal_DelayedFault] TO [Proficydbo]
GO
```

## Model 200

The downtime detection models are the logic that determines when the line is running and when downtime is occurring. It also can translate available signals from the control system into fault values and associate reasons.

Model 200 compares the value of the Input Tag to the Running Value. If the tag is equal to the Running Value, the line is considered running. If it is not equal to the Running Value, then it is considered down. If the line is determined to be down, the value of the Input Tag can be used to lookup the fault in the fault list and associate reasons.

The information from the **PT:ProdRateTag** property and the **Prod Rate Conv Info** property is stored in the Timed\_Event-Details table in Production\_Event.

---

*The functionality in this model is available only if you have purchased the Efficiency Module.*

---

### Model 200 Properties






The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| PT:Delay Tag                      | The value of this tag determines the current fault. If the value of the tag is equal to the Running Value, then the equipment is considered to be running. If the value of the tag is not equal to the Running Value, the value of the tag is used to look up the fault in the fault list.                 |
| Running Value                     | The value of this tag indicates that the equipment is running. All other values create downtime.   |
| PT:ProdRateTag                    | Select the historian tag that indicates your production rate, such as bottles per day.   |
| Prod Rate Conv. Info              | Enter the value that will convert your production rate into minutes. For example, if your production rate is "per day", then enter "1440."   |
| Minimum Delay Time (Secs)         | The value of this tag indicates the minimum length of downtime event to log. Downtime events that last only "n" seconds are ignored.   |


## Model 210






Model 210 detects downtime on a single location only. The location is either running or not running, depending on the VBScript that you enter. To use this model, you must have only one location defined.

To detect downtime on a single location:

1. In the  Plant Model, right-click the production unit where you want to detect events and click **Configure Events on <production unit>**. The **Event Detection** wizard appears.
2. On the **Configured Models** tab, select **Downtime** from the **Model Type** list and click **Add New Model**. The **General**, **Reason Tree Configuration**, **Identify Input(s)**, and **Scripts** tabbed pages appear.
3. Click the **General** tab and do the following:
  - a. Select **Fault on a Single Location (210)**.
  - b. **optional:** In the Maximum Run Time (Seconds) box, type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop).
  - c. **optional:** Type the group number in the Model Processing Group box. The Model Processing Group number can be up to six digits.
  - d. **optional:** In the **Extended Information** box, type additional information that may be useful. This is not used on any reports or displays.
  - e. **optional:** In the **Exclusions** box, type the values this model is to ignore and not use to trigger the model. Use a comma to separate values.
  - f. To require an electronic signature, select the level of authorization from the **Signature Level** list. For more information on electronic signatures, please see Using Electronic Signatures.
  - g. From the **FaultMode** list, select one of the following fault modes.
    - **Assign fault at start change = split:** The fault is assigned immediately when a new downtime record is opened. If a new fault is passed while the initial downtime record is still open, the downtime records are chained together and the assigned fault is passed in as a split.
    - **Assign Fault @ Start:** Assign the fault code when the downtime record is opened.
    - **Assign Fault @ End:** Assign the fault code when the downtime record is closed.
    - **Overwrite fault at End Time:** Assign the fault code at the beginning of the downtime event when opened and if the fault code changes when the record is closed, the downtime record will be updated to reflect the new fault code.
  - h. Select **Automatically Add Missing Faults** to add faults if the fault from the fault tag does not exist in the Plant Applications database.
5. Click the **Reason Tree Configuration** tab and do the following:
  - a. Click the **Downtime Locations** tab to assign reason trees to the downtime event.
    - Select the production unit. The **Tree Selection** dialog box appears and three buttons are displayed on the toolbar:  **Assign Cause Tree**,  **Assign Action Tree**, and  **Enable Research**.
    - Select a reason tree and click OK. The selected reason tree is listed under the **Cause Reason Tree** column.
    - Click  **Assign Action Tree** to assign an action reason tree. The **Tree Selection** dialog box appears.
  - Select a reason tree and click **OK**. The selected reason tree is listed under the **Action Reason Tree** column

---

**NOTE:** If you need to create a new action or cause reason tree, click  **Manage Trees**. The **Tree Builder** dialog box appears where you can create new reason trees.



- **optional:** Click  **Enable Research**. **Enabled** is displayed under the **Enable Research** column. Enable Research will enable the Research tab in the Sequence of Events display for downtime events. Enabling research allows the user to identify a site user as the person responsible for researching the downtime event and to set the research as closed or open.
  - b. Click the **Fault Translation** tab to create and assign a fault to the downtime event.
    - Click Add. A new row is added under Fault Translation For Detection Model Output.
    - In the Fault column, type a fault value. The fault value must match a value that will be returned by an historian tag (if using model 200) or a fault value identified in a script (if using model 210 or 211). Fault values must be unique on this production unit.
    - In the Fault Name column, type a name for the new fault. Fault names must be unique on this production unit.
    - Select the production unit from the Location list. If the production unit has slave units configured, the production unit and its slave units will be available in the Location list.
    - In the Reason1 column select a reason from the list. The contents on the list will depend on the reason tree selected on the Cause Reason Trees tab. Depending on how the reason tree is configured, you may select subsequent reasons for Reason2 through Reason4.
  - c. Click the **Reason Shortcuts** tab to create a shortcut that will be available in the downtime display. When the reason shortcut is selected in the downtime display, reasons are automatically added to the downtime event.
    - Click  **Insert Input**. A new row is added.
    - Under **Shortcut Name**, type the name of the shortcut. This name will be displayed on the right-click menu in the downtime display.
    - Under **Time**, type the duration, in minutes, of the downtime event. When the shortcut is applied to a downtime event in a downtime display, this will become the amount of downtime for the event.
    - Under **Location**, select the production unit.
    - Under **Reason Level 1 – 4**, select the reason levels to apply to the downtime event in a downtime display. The available reasons are determined by the selected cause reason tree.
  - d. Click the **Status Translation Table** tab to create a list that is displayed for selection in the downtime display.
    - Click  **Insert Input**. A new row is added.
    - Under **Status Name**, type the name of the status. This will be available from the Status list in the downtime display.
    - Under **Status Value**, type a numeric value for the status. The status value acts as an index value for the status and can be used in custom stored procedures; it is not displayed anywhere.
- 6. Click the **Identify Input(s)** tab to identify and select the historian tags used for input.
  - a. To determine if the unit is down select an historian tag for the **RunTag** Tag box.
  - b. To determine the location of the downtime select **Set Downtime Location**. The **LocTag** row appears.
  - c. To determine the fault or root cause, select **Set Downtime Fault**. The **FaultTag** row appears.
  - d. For each row, click  or . The **Tag Search** dialog box appears.

- e. Select a tag and click **OK**.

---

*NOTE: You must select at least one tag as the trigger tag. For more information, see [Input tags](#).*


---





7. Click the **Scripts** tab to modify the sample scripts, or click  and type your own scripts.
- For more information and to view two examples of running logic, see [Running Logic](#).
  - For more information and to view two examples of fault logic, see [Fault Logic](#).
8. Click  to activate the model.

## Model 211


Model 211 determines downtime by checking various locations to determine if the production line is up or down. It determines the location based on input tag values.

To determine cause location from inputs:




1. In the  Plant Model, right-click the production unit where you want to detect events and click **Configure Events on <production unit>**. The **Event Detection** wizard appears.
2. On the Configured Models tab, select Downtime from the Model Type list and click Add New Model. The General, Reason Tree Configuration, Identify Input(s), and Scripts tabbed pages appear.
3. On the **General** tabbed page, do the following:
  - a. Select Cause Location From Inputs (211).
  - b. optional: In the Maximum Run Time (Seconds) box, type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop).
  - c. optional: Type the group number in the Model Processing Group box. The Model Processing Group number can be up to six digits.
  - d. optional: In the Extended Information box, type additional information that may be useful. This is not used on any reports or displays.
  - e. optional: In the Exclusions box, type the values this model is to ignore and not use to trigger the model. Use a comma to separate values.
  - f. To require an electronic signature, select the level of authorization from the Esignature Level list. For more information on electronic signatures, please see [Using Electronic Signatures](#).
  - g. From the FaultMode list, select one of the following fault modes.
    - **Assign fault at start change = split:** The fault is assigned immediately when a new downtime record is opened. If a new fault is passed while the initial downtime record is still open, the downtime records are chained together and the assigned fault is passed in as a split.
    - **Assign Fault @ Start:** Assign the fault code when the downtime record is opened.
    - **Assign Fault @ End:** Assign the fault code when the downtime record is closed.
    - **Overwrite fault at End Time:** Assign the fault code at the beginning of the downtime event when opened and if the fault code changes when the record is closed, the downtime record will be updated to reflect the new fault code.
  - h. Select **Automatically Add Missing Faults** to add faults if the fault from the fault tag does not exist in the Plant Applications database.









4. On the **Reason Tree Configuration** tabbed page, do the following:
  - a. On the **Downtime Locations** tabbed page do one or more of the following:
    - Select the production unit. The **Tree Selection** dialog box appears and three buttons are displayed on the toolbar:  **Assign Cause Tree**,  **Assign Action Tree**, and  **Enable Research**.
    - Select a reason tree and click OK. The selected reason tree is listed under the **Cause Reason Tree** column.
    - Click  **Assign Action Tree** to assign an action reason tree. The **Tree Selection** dialog box appears.
    - Select a reason tree and click **OK**. The selected reason tree is listed under the **Action Reason Tree** column

---

*NOTE: If you need to create a new action or cause reason tree, click  **Manage Trees**. The **Tree Builder** dialog box appears where you can create new reason trees.*

---


- **optional:** Click  **Enable Research**. **Enabled** is displayed under the **Enable Research** column. Enable Research will enable the Research tab in the Sequence of Events display for downtime events. Enabling research allows the user to identify a site user as the person responsible for researching the downtime event and to set the research as closed or open.
- b. On the **Fault Translation** tabbed page, do the following:
    - Click Add. A new row is added under Fault Translation For Detection Model Output.
    - In the Fault column, type a fault value. The fault value must match a value that will be returned by an historian tag (if using model 200) or a fault value identified in a script (if using model 210 or 211). Fault values must be unique on this production unit.
    - In the Fault Name column, type a name for the new fault. Fault names must be unique on this production unit.
    - Select the production unit from the Location list. If the production unit has slave units configured, the production unit and its slave units will be available in the Location list.
    - In the Reason1 column select a reason from the list. The contents on the list will depend on the reason tree selected on the Cause Reason Trees tab. Depending on how the reason tree is configured, you may select subsequent reasons for Reason2 through Reason4.
  - c. On the **Reason Shortcuts** tabbed page, do the following:
    - Click  **Insert Input**. A new row is added.
    - Under **Shortcut Name**, type the name of the shortcut. This name will be displayed on the right-click menu in the downtime display.
    - Under **Time**, type the duration, in minutes, of the downtime event. When the shortcut is applied to a downtime event in a downtime display, this will become the amount of downtime for the event.
    - Under **Location**, select the production unit.
    - Under **Reason Level 1 – 4**, select the reason levels to apply to the downtime event in a downtime display. The available reasons are determined by the cause reason tree selected in step 5a.
  - d. On the **Status Translation Table** tabbed page, do the following:
    - Click  **Insert Input**. A new row is added.





- Under **Status Name**, type the name of the status. This will be available from the Status list in the downtime display.
  - Under **Status Value**, type a numeric value for the status. The status value acts as an index value for the status and can be used in custom stored procedures; it is not displayed anywhere.
5. On the **Identify Input(s)** tabbed page, do the following:
- To determine if the unit is down:
    - a. Select the **RunTag** Tag box and click  or . The **Tag Search** dialog box appears.
    - b. Select a tag and click **OK**.
    - c. If you want this tag to be a trigger tag, select the Trigger check box.
  - To determine the location of the downtime:
    - a. Select **Set Downtime Location**. The LocTag row appears.
    - b. Select the **Tag** box and click  or . The **Tag Search** dialog box appears.
    - c. Select a tag and click **OK**.
    - d. If you want this tag to be a trigger tag, select the Trigger check box.
  - To determine the fault or root cause:
    - a. Select **Set Downtime Fault**. The FaultTag row appears.
    - b. Select the **Tag** box and click  or . The **Tag Search** dialog box appears.
    - c. Select a tag and click **OK**.
    - d. If you want this tag to be a trigger tag, select the Trigger check box.
- 
- NOTE: You must select at least one tag as the trigger tag. For more information, see [Input tags](#).*
6. On the **Scripts** tabbed page, fault logic, location logic and running logic scripts are automatically generated. However, you can modify the scripts, or you can click  and type your own scripts.
- For more information and to view two examples of running logic, see [Running Logic](#).
  - For more information and to view two examples of fault logic, see [Fault Logic](#).
  - For more information on location logic, see [Model 211 Location Logic](#).
7. Click  to activate the model.

## Model 212


Model 212 determines the cause location by defining the states of each location (or piece of equipment) along the line. Based on how you define the state of each location, the downtime detection model (as opposed to the control system) attempts to determine which piece of equipment along a line was the source or cause of downtime.

To determine cause location from equipment states:








1. In the  Plant Model, right-click the production unit where you want to detect events and click **Configure Events on <production unit>**. The **Event Detection** wizard appears.

2. On the **Configured Models** tab, select **Downtime** from the **Model Type** list and click **Add New Model**. The **General**, **Reason Tree Configuration**, **Identify Input(s)**, and **Scripts** tabbed pages appear.
3. On the **General** tabbed page, do the following:
  - a. Select Cause Location From Equipment States (212).
  - b. **optional:** In the **Maximum Run Time (Seconds)** box, type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop).
  - c. **optional:** Type the group number in the Model Processing Group box. The Model Processing Group number can be up to six digits.
  - d. **optional:** In the **Extended Information** box, type additional information that may be useful. This is not used on any reports or displays.
  - e. **optional:** In the **Exclusions** box, type the values this model is to ignore and not use to trigger the model. Use a comma to separate values.
  - f. To require an electronic signature, select the level of authorization from the **Esignature Level** list. For more information on electronic signatures, please see [Using Electronic Signatures](#).
  - g. From the FaultMode list, select one of the following fault modes.
    - **Assign fault at start change = split:** The fault is assigned immediately when a new downtime record is opened. If a new fault is passed while the initial downtime record is still open, the downtime records are chained together and the assigned fault is passed in as a split.
    - **Assign Fault @ Start:** Assign the fault code when the downtime record is opened.
    - **Assign Fault @ End:** Assign the fault code when the downtime record is closed.
    - **Overwrite fault at End Time:** Assign the fault code at the beginning of the downtime event when opened and if the fault code changes when the record is closed, the downtime record will be updated to reflect the new fault code.
  - h. Select **Automatically Add Missing Faults** to add faults if the fault from the fault tag does not exist in the Plant Applications database.
4. On the **Reason Tree Configuration** tabbed page, do the following:
  - a. On the **Downtime Locations** tabbed page do one or more of the following:
    - Select the production unit. The **Tree Selection** dialog box appears and three buttons are displayed on the toolbar:  **Assign Cause Tree**,  **Assign Action Tree**, and  **Enable Research**.
    - Select a reason tree and click OK. The selected reason tree is listed under the **Cause Reason Tree** column.
    - Click  **Assign Action Tree** to assign an action reason tree. The **Tree Selection** dialog box appears.
    - Select a reason tree and click **OK**. The selected reason tree is listed under the **Action Reason Tree** column

---



*NOTE: If you need to create a new action or cause reason tree, click  **Manage Trees**. The **Tree Builder** dialog box appears where you can create new reason trees.*

---

- **optional:** Click  **Enable Research**. **Enabled** is displayed under the **Enable Research** column. Enable Research will enable the Research tab in the Sequence of Events display for downtime events. Enabling research allows the user to identify a site user as the person responsible for researching the downtime event and to set the research as closed or open.
- b. On the **Fault Translation** tabbed page, do the following:
  - Click Add. A new row is added under Fault Translation For Detection Model Output.
  - In the Fault column, type a fault value. The fault value must match a value that will be returned by an historian tag (if using model 200) or a fault value identified in a script (if using model 210 or 211). Fault values must be unique on this production unit.
  - In the Fault Name column, type a name for the new fault. Fault names must be unique on this production unit.
  - Select the production unit from the Location list. If the production unit has slave units configured, the production unit and its slave units will be available in the Location list.
  - In the Reason1 column select a reason from the list. The contents on the list will depend on the reason tree selected on the Cause Reason Trees tab. Depending on how the reason tree is configured, you may select subsequent reasons for Reason2 through Reason4.
- c. On the **Reason Shortcuts** tabbed page, do the following:
  - Click  **Insert Input**. A new row is added.
  - Under **Shortcut Name**, type the name of the shortcut. This name will be displayed on the right-click menu in the downtime display.
  - Under **Time**, type the duration, in minutes, of the downtime event. When the shortcut is applied to a downtime event in a downtime display, this will become the amount of downtime for the event.
  - Under **Location**, select the production unit.
  - Under **Reason Level 1 – 4**, select the reason levels to apply to the downtime event in a downtime display. The available reasons are determined by the cause reason tree selected in step 5a.
- d. On the **Status Translation Table** tabbed page, do the following:
  - Click  **Insert Input**. A new row is added.
  - Under **Status Name**, type the name of the status. This will be available from the Status list in the downtime display.
  - Under **Status Value**, type a numeric value for the status. The status value acts as an index value for the status and can be used in custom stored procedures; it is not displayed anywhere.
- 5. On the **Identify Input(s)** tabbed page, do the following:
  - To determine if the unit is down:
    - a. Select the **RunTag** Tag box and click  or . The **Tag Search** dialog box appears.
    - b. Select a tag and click **OK**.
    - c. If you want this tag to be a trigger tag, select the Trigger check box.
  - To determine the location of the downtime:
    - a. Select **Set Downtime Location**. The LocTag row appears.
    - b. Select the **Tag** box and click  or . The **Tag Search** dialog box appears.
    - c. Select a tag and click **OK**.





## Models

- d. If you want this tag to be a trigger tag, select the Trigger check box.
- To determine the fault or root cause:
  - a. Select **Set Downtime Fault**. The FaultTag row appears.
  - b. Select the **Tag** box and click  or . The **Tag Search** dialog box appears.
  - c. Select a tag and click **OK**.
  - d. If you want this tag to be a trigger tag, select the Trigger check box.

---

*NOTE: You must select at least one tag as the trigger tag. For more information, see [Input tags](#).*

---

6. On the **Scripts** tabbed page, fault logic and running logic scripts are automatically generated. However, you can modify the scripts, or you can click  and type your own scripts.
  - For more information and to view two examples of running logic, see [Running Logic](#).
  - For more information and to view two examples of fault logic, see [Fault Logic](#).
  - For more information on location logic, see [Model 212 Location Logic](#).
7. Click  to activate the model.

## Model 211 Location Logic

---

*The functionality in this model is available only if you have purchased the Efficiency Module.*

---

### Establish Location Logic for the Location Determined from Inputs (Model 211)

Both the **Location** and **Equipment State** drop-down boxes are disabled, and you need to define only a single script. Using the variable data available, determine which location is the source location and end the script with `Location = <location name>`. You can search variables and automatically insert them into the script at the current cursor position by selecting `Insert Input...` and selecting the appropriate input. You can also search location names by clicking `Insert Location...` and selecting the appropriate location.

If no location can be determined, set `Location = ""`

#### Location Logic Example #1 (Model 211)

In this example, if A is non-zero then Robot 1 is said to be the location; if B is non-zero then Robot 2 is said to be the location, etc.

```
If A then Location = "Robot 1"  
If B then Location = "Robot 2"  
If C then Location = "Robot 3"
```

#### Location Logic Example #2 (Model 211)

In this example, the alias A is examined to see which location is the cause:

```
If A = 1 then Location = "Robot 1"  
If A = 2 then Location = "Robot 2"  
If A = 3 then Location = "Robot 3"
```

## Model 212 Location Logic

---

*The functionality in this model is available only if you have purchased the Efficiency Module.*

---

## Establishing Location Logic for the Cause Location Determined By Defining Equipment States (Model 212)

For each location and state, you must define whether the status for that particular location and state is true or false. If you do not define the status of any locations or states, they are assumed to be false.

The possible states of a location on a production line are unavailable, down, starved, and blocked. Under this model, the source location is determined by the following logic:

- Each location is scanned based on order in the production line.
- Each state is scanned in the order listed above.
- Unavailable locations are ignored, as are locations without any logic embedded in them. Locations without unavailable scripts are assumed to be False and therefore will be included in the scan of the other states.
- The first location with a down state is considered the source location.

If no locations are down, the location immediately preceding the first starved location is considered the source location. If the first location is starved, it is considered the source location.

If no locations are starved, the location immediately following the first blocked location is considered the source location. If the last location is blocked, it is considered the source location.

If no locations are down, starved, or blocked, then no source location is set.

Under this model, the function for each location or state must end with Status = True or Status = False. Input variables are referenced using their alias (for example, A, B, C, D, and so forth). You can search variables and automatically insert them into the script at the current cursor position by clicking the Insert Input... button and selecting the appropriate input.

### Location Logic Example #1 (Model 212)

This is an example of a "Starved" location logic script for a location named "Robot 1". When Status = True, this location is said to be starved (where this location is Robot 1). In this case, we first round the number to an integer before checking to see if it is = 2.

```
if Round(A) = 2 then
Status = True
else
Status = False
end if
```

### Location Logic Example #2 (Model 212)

This is an example of a "Down" location logic script for a location named "Robot 1". When Status = True, that means that this location is said to be down (where this location is Robot 1). In this case, if the value is less than 2, it is said to be down.

```
If A < 2 then
Status = True
else
Status = False
end if
```

## Export Models

## Model 72

Model 72 exports Roll data after every Interval is elapsed. The data is exported to an external database using an ODBC data source.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

**IMPORTANT:** *The ODB connection must NOT have the parameter "Enable Lazy Close Support" selected to avoid locking the Plant Applications tables.*

---

### Model 72 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                                 | Description  |
|--|--|
| <b>Maximum Run Time (Seconds)</b>        | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group                   | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:(Interval) Minutes                  | A value change of this tag is used to trigger a new production event to be created.  |
| ODBC Connect String                      | DSN=xx;UID=yyy; PWD=zzz;   |
| Warehouse (Ex. LA)                       |  |
| StoreFunc (Ex. GBK_Store_Released_Roll ) |  |
| Local Get Data spName                    |  |
| Local Confirm Data spName                |  |
| Get Status Id (Ex. 9)                    |  |
| Confirm Status Id (Ex. 8)                |  |
| Hold Status Id (Ex. 13)                  |  |
| StatusFunc (Ex. GBK_Get_Roll_Status )    |  |

## Model 73

Model 73 imports EWMA data after every Interval is elapsed.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 73 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:(Interval) Minutes           | A value change of this tag is used to trigger a new production event to be created.  |

## Model 77 – Data Export

Model 77 exports data after the specified interval elapses. At the interval specified by the value in the TINT:Interval field, Model 77 fires the stored procedure, creates a file and saves it to the location specified in the Transfer File Path field. A file is created for each row that is returned within the interval. For example, if the specified interval is 5 minutes and 10 events occur in the 5 minutes since the model was last fired, then 10 files will be created.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 77 Properties

The following properties are defined using the Event Configuration wizard.

| Property                                  | Description  |
|---|--|
| <b>Maximum Run Time (Seconds)</b>         | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group                    | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| <b>Extended Information</b>               | Additional information that may be useful. This is not used on any reports or displays.  |
| <b>Exclusions</b>                         | The values this model is to ignore and not use to trigger the model.   |
| <b>TINT:Interval (Minutes)</b>            | How often the stored procedure is executed, in minutes. For example, entering 5 will cause the stored procedure to run every five minutes.   |
| <b>Construction File</b>                  | The location where the pre-processed files are stored. The file name must be <b>? .txt</b> . The question mark (?) is replaced with the value of the first field in the stored procedure. In the sample stored procedure, the name of the file would be the value of event_num.                            |
| <b>Transfer File Path</b>                 | Directory to put the processed file.   |
| <b>Local spName (Ex. spLocal_GetData)</b> | Stored procedure used to build the custom file. The stored procedure name must start with <b>spLocal_</b> . The stored procedure requires no input parameters, but does require three output parameters which are used to create the export file.  |

### Sample Stored Procedure

```

Declare @ResultSet Table(
  EventNum VarChar(20) Not Null,
  CommentId Int,
  myValue VarChar(255)
)
Declare @Now datetime
select @Now = getdate()
INSERT INTO @ResultSet (EventNum, CommentId,myValue)
      Select event_Num,Comment_Id,Event_Status
      From events where timestamp between dateadd(minute,-25,@now) and @Now
and pu_Id = 22
select EventNum,CommentId,myValue from @ResultSet

```

## Model 78

## Models

Model 78 exports data after the specified interval elapses. At the interval specified by the value in the TINT:Interval field, Model 78 fires the stored procedure, creates a file and saves it to the location specified in the Transfer File Path field. A file is created for each row that is returned within the interval. For example, if the specified interval is 5 minutes and 10 events occur in the 5 minutes since the model was last fired, then 10 files will be created.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 78 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                                  | Description  |
|---|--|
| <b>Maximum Run Time (Seconds)</b>         | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group                    | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| <b>Extended Information</b>               | Additional information that may be useful. This is not used on any reports or displays.  |
| <b>Exclusions</b>                         | The values this model is to ignore and not use to trigger the model.   |
| <b>TINT:Interval (Minutes)</b>            | How often the stored procedure is executed, in minutes. For example, entering 5 will cause the stored procedure to run every five minutes.   |
| <b>Construction File</b>                  | The location where the pre-processed files are stored. The file name must be <b>? .txt</b> . The question mark (?) is replaced with the value of third parameter in the stored procedure. In the sample stored procedure, the file name would be the value of the event number.                            |
| <b>Transfer File Path</b>                 | Directory to put the processed file.   |
| <b>Local spName (Ex. spLocal_GetData)</b> | Stored procedure used to build the custom file. The stored procedure name must start with <b>spLocal_</b> . The stored procedure requires no input parameters, but does require 11 output parameters which are used to create the export file.   |

### Sample Stored Procedure

```
Declare @ResultSet Table(
  ProdLine VarChar(255) Not Null,
  ProdUnit VarChar(20) Not Null,
  EventNum VarChar(20) Not Null,
  TimeYear Int Not Null,
  TimeMonth Int Not Null,
  TimeDay Int Not Null,
  TimeHour Int Not Null,
  TimeMinute Int Not Null,
  TimeSecond Int Not Null,
  TestName VarChar(20) Not Null,
  TestValue VarChar(20) Not Null
)

INSERT INTO @ResultSet
(ProdLine,ProdUnit,EventNum,TimeYear,TimeMonth,TimeDay,TimeHour,TimeMinute,
TimeSecond,TestName,TestValue)
```

```

Select
'MyLine',pu_id,event_num,Datepart (Year, Timestamp) ,Datepart (Month, Timestamp)
,Datepart (Day, Timestamp) ,Datepart (hour, Timestamp) ,Datepart (Minute, Timestamp
) ,Datepart (Second, Timestamp) , 'MyTest' , 555

From events

select
ProdLine,ProdUnit,EventNum,TimeYear,TimeMonth,TimeDay,TimeHour,TimeMinute,T
imeSecond,TestName,TestValue from @ResultSet

```

## Model 92

Model 92 exports Reel data after every Interval is elapsed. The data is exported to a file into the Transfer File Path defined in the model properties. Custom Stored Procedures are used to create the files. This model is set up for a specific site.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 92 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property   | Description  |
|--|--|
| <b>Maximum Run Time (Seconds)</b>                  | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group                             | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)                            | Timing Interval the model checks the table for new files to export.  |
| Local Get Events spName (Ex. spLocal_GetEvents)    | Stored Procedure used to retrieve Event data.  |
| Local Get Set Data spName (Ex. spLocal_GetSetData) | Stored Procedure used to retrieve Set data.  |
| Construction File Path                             | Directory used to build the file.  |
| Transfer File Path                                 | Directory to put the completed file.   |

## Model 98

Model 98. This model is set up for a specific site.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 98 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| spName                            |  |
| Status VarId                      | The variable that contains the Status.   |

## Models

|              |  |
|--------------|--|
| Misc Varld 1 |  |
| Misc Varld 2 |  |

## Model 99

Model 99 exports the Event Status and Event Comments for Reels after every Interval is elapsed. The data is exported to a file into the Transfer File Path defined in the model properties. A custom Stored Procedure is used to create the files. This model is set up for a specific site.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 99 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks the table for new files to export.  |
| spName                            | Stored Procedure used to retrieve Event data.  |
| Construction File Path            | Directory used to build the file.  |
| Transfer File Path                | Directory to put the completed file.   |
| Record Type                       |  |

## Genealogy Models 1051-1055

These are recommended Generic Models to use in Production Management and Genealogy. They offer flexibility to configure individual sites.

## Generic Models 600-603

Generic Models 600, 601, 602, and 603 are used to trigger the calling of a custom SQL Stored Procedure. The custom Stored Procedure allows the flexibility to use Result Sets to create, update, and delete data in the Plant Applications database. See the section on Result Sets for further explanation of their features.

### Generic Model Description and Usage

All Plant Applications models have an assigned event type that is used by the Administrator to group models together (waste, downtime, production event, etc.). Generic models are a special class of models that are designed to be flexible and are never actually directly configured. Instead they are used as a template for models that are created for a more specific purpose. Generic models can be used to create models of almost any type; they can even create models that create events of many types. For example, a common use of generic models is to build a model that produces both production events and product change events.

In order to use a generic model, you must first create your own model with a specific event type. Even if you use a generic-based model to produce multiple types of events, only a single event type can and needs to be specified. In the example above, it would be common to create this generic based

model with an event type of production event. Assigning an event type to your generic based model is important because this allows it to be configured through the Event Configuration dialog in the Administrator.

To configure a Generic Model:

1. In the Plant Applications Administrator, click **Global Configuration**.
2. Right-click on **Administer Models** and select **Administer Models** from the pop-up menu.
3. In the **Event Detection Model Manager** dialog box, select **Generic** from the **Event Type** drop-down list and then click the **Search** button.
4. Select a model template and then click the **New** button.
5. The **Model General** tab will be displayed. Note the new model number. All models generated through this process are given a model number greater than 50,000. Use the fields on this tab to document your model (name, version, comments, etc.). Most importantly on this tab, make sure to select the correct Event Type (the default is Production Event) because this will determine where this new model is configured later. Select the **Model Properties** tab to specify the properties used by this new model. Some properties may be required; if so they will already be listed. Some models allow the addition of other properties that are optional to the Generic template. In addition to adding comments to properties, use this tab to set defaults and make properties that are optional or locked at configuration time. For properties that allow it, the number of instances of a selected property can be set.
6. Select the **User-defined Properties** tab to add additional properties that are not part of the generic model template. At run time, the stored procedure associated with this new model can look up these user-defined properties from that Event\_Configuration table based on the EC\_Id which is sent in to most generic-based models.

The custom Stored Procedure allows the flexibility to use the Result Sets to create, update, and delete data in Plant Applications.

[Click here for more information about using Result Sets in models.](#)

## Generic Model 600-Transition Triggered

Generic Models 600, 601, 602, and 603 are used to trigger the calling of a custom SQL Stored Procedure. The custom stored procedure allows the flexibility to use result sets to create, update, and delete data in the Plant Applications database. See the section on Result Sets for further explanation of their features.

### Model 600 Description

Model 600 is triggered by historian tag value transitions from a specific value to another specific value. When a Historian Tag value changes, the model calls the specified stored procedure. The stored procedure has specific parameters that must be defined. The model passes data into those parameters and receives a message indicating success or failure. Historian values can also be passed into the stored procedure. The sampling type is a numeric value and can be found in the Sampling\_Type table in the Plant Applications SQL database. Result Set queries are created in SQL and the Event Manager service handles these Result Sets as real time messages in Plant Applications. For more information, see the Result Sets chapter.

### Model 600 Properties

The following model properties are set up using the Plant Applications Administrator:

| Property                          | Description   |
|-----------------------------------|---|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time |



|                                |  |
|--------------------------------|--|
|                                | out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group         | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TriggerTag                     | Tag used to trigger the calling of the stored procedure.   |
| From Value                     | The value the trigger tag must transition FROM to trigger the creation of a new production event.  |
| To Value                       | The value the trigger tag must transition TO in order to trigger the creation of a new production event.   |
| Local SP Name                  | Stored procedure called when the model is triggered.   |
| Tag#1 (Optional)               | Tag#1 used to pass values into stored procedure.   |
| Tag#1 Sampling Type (Optional) | The sampling type used to acquire a value for Tag#1. The sampling type is a numeric value and can be found in the Sampling_Type table.                                 |
| Tag#.....(Optional)            | Values can be retrieved for up to 100 tags (optional)  |

### Model 600 Stored Procedure Parameters

The following parameters are required at the beginning of the stored procedure that is called by Model 600. The stored procedure must have the same input and output parameters as defined and the parameter variables can be named whatever is desired but the order must be maintained. Up to 100 Historian Tag values can be passed into the Stored Procedure.

| SQL Variable Name                      | Parameter Description   |
|--|---|
| @ReturnStatus int OUTPUT,              | Flag to indicate Success or Failure (1-Success,0-Failure)   |
| @ReturnMessage<br>varchar(255) OUTPUT, | Error Message to Write to Log File  |
| @EC_Id int,                            | Input to the Stored Procedure indicating the EC_Id of the model from the Event_Configuration table. |
| @Timestamp datetime,                   | The New Time of the Triggering Tag  |
| @Hist1Value varchar(30),               | The Value For Tag 1 (Optional)  |
| @Hist2Value varchar(30),               | The Value For Tag 2 (Optional)  |
| @Hist3Value varchar(30),               | The Value For Tag 3 (Optional)  |
|  | Values can be retrieved for up to 100 Tags (Optional)   |

To control how far back in time the Event Manager service looks for tag changes after a reload or restart of the Event Manager service, you can use Result Set 51 in your stored procedure. In the result set, use 'JumpToTime' for the ParameterName and for the ParameterValue, use the timestamp for how far back in time you want to look for tag changes. For more information, see the topic Result Set 51 in the online help.

### Setting Up a Generic Model 600

1. [Create an Event Type](#) that gives context and information to the type of event to be created. This can be any event type available in the system. If it is a Production Event type, then specific information about the dimensions of the product being made are specified in this placeholder Event Type.
2. [Create a Model Template](#) derived from Model 600 (or other Generic Models).
3. [Assign new Model](#) and specific Model parameter values to be used with the new Model Template derived from Model 600 (or other Generic Models). Specific parameters would be things like Stored Procedure Name, Variables, and Variable sampling Types.

### Add a Specific Event Type such as a Production Event Type

1. Browse to the "Plant Model" and go inside the Production Line.

2. Select the Production Unit; right-click and select "Configure Events"
3. From the "Event Configuration" window, browse through the "Available Events To Add," locate the appropriate Event of type "Production Event".
4. Select the "Add Event" button and the Production Event will be added to the selected Production Unit.
5. If A Production Event of the proper type with the exact dimensions does not exist, add a new Production Event Type by selecting the "Manage Subtypes" button. Existing Event Types can be reviewed by Editing an Event using the Master List as well.
6. Select "Production Event" in the "Type To Add" dropdown and select "Add Event". This Production Event can be used as the placeholder production event that is used to indicate the specific dimensions to be tracked for the product being made.
7. Fill in the name of the Production Event, and its primary dimension ("Dimension X") information. The primary dimension used to measure production and can be used as a placeholder event if required. Enable one or more of the other dimensions (Y, Z, A) and enter their name and engineering units. These dimensions are used to track dimensional type products (those with length and width).
8. When finished, select "OK" to add the new Production Event Type to the Master List. **Reminder:** This event is used as a placeholder event to give context to the product being made on this unit.
9. With the new Production Event type added to the list, close the Master Event List window and return to the Event Configuration window.
10. Select "Add Event" to add the new Production Event as an Event Type on this specific Production Unit.

#### **Create a Model Template derived from Model 600**

1. Select the Production Event added in the **Events Enabled On This Unit** in the Event Configuration window.
2. Select the **Configure Event Detection Models** tab.
3. Select the Master List button in the lower left corner. Select Model Number 600 from the list and hit the New button to create a new Template Model that will be derived from Model 600.
4. On the **Model General** tab, select Production Event as the Event Type. The Name of the Model will be the same as the Model it is derived from. This name can be edited if desired. The number for the Model is a new number that is arbitrarily assigned by the system. All new Models derived from Generic Models will start at 50,000 and count up.
5. Select the **Model Properties** tab. The default Fields available are listed. If more Tag values are required to be retrieved into the Stored Procedure then these additional parameters can be added by selecting the **New** button.
6. Close this window to complete building the new Model Template. In this example the new Model Number assigned is 50654.

#### **Assign New Model and the Specific Model Property Values**

1. Click on the Production Event from the upper window called **Events Enabled On This Unit**.
2. In the Available Models to Assign window, find the Model Number 50654 and hit the **Assign Model** button.
3. Specify each required Model Property for the newly assigned Model in the lower right window **Selected Model Properties**. Select each Model Property in the and select **Edit**.
4. A dialog appropriate for the Data Type of each Property will appear.
5. Select an appropriate item from the list presented, or type in an appropriate Value and select "OK".

## Models

6. To Activate the selected Model, hit the **Activate** button. Models with required Properties blank cannot be activated.
7. Reload the Event Manager.

## Setting Up a Generic Model 601

1. [Create an Event Type](#) that gives context and information to the type of event to be created. This can be any event type available in the system. If it is a Production Event type, then specific information about the dimensions of the product being made are specified in this placeholder Event Type.
2. [Create a Model Template](#) derived from Model 601 (or other Generic Models).
3. [Assign new Model](#) and specific Model parameter values to be used with the new Model Template derived from Model 601 (or other Generic Models). Specific parameters would be things like Stored Procedure Name, Variables, and Variable sampling Types.

### Add a Specific Event Type such as a Production Event Type

1. Browse to the "Plant Model" and go inside the Production Line.
2. Select the Production Unit; right-click and select "Configure Events"
3. From the "Event Configuration" window, browse through the "Available Events To Add," locate the appropriate Event of type "Production Event".
4. Select the "Add Event" button and the Production Event will be added to the selected Production Unit.
5. If A Production Event of the proper type with the exact dimensions does not exist, add a new Production Event Type by selecting the "Manage Subtypes" button. Existing Event Types can be reviewed by Editing an Event using the Master List as well.
6. Select "Production Event" in the "Type To Add" dropdown and select "Add Event". This Production Event can be used as the placeholder production event that is used to indicate the specific dimensions to be tracked for the product being made.
7. Fill in the name of the Production Event, and its primary dimension ("Dimension X") information. The primary dimension used to measure production and can be used as a placeholder event if required. Enable one or more of the other dimensions ("Y","Z","A") and enter their name and engineering units. These dimensions are used to track dimensional type products (those with length and width).
8. When finished, select "OK" to add the new Production Event Type to the Master List.

---

*This event is used as a placeholder event to give context to the product being made on this unit.*

---

9. With the new Production Event type added to the list, close the Master Event List window and return to the Event Configuration window.
10. Select "Add Event" to add the new Production Event as an Event Type on this specific Production Unit.
11. This completes the setup of the place holder event. The place holder event is the Production Event that contains the Name and Dimension Types to be manufactured.

### Create a Model Template derived from Model 601

1. Select the Production Event added in the **Events Enabled On This Unit** in the Event Configuration window.
2. Select the **Configure Event Detection Models** tab.
3. Select the Master List button in the lower left corner. Select Model Number 601 from the list and hit the New button to create a new Template Model that will be derived from Model 601.

4. On the **Model General** tab, select Production Event as the Event Type. The Name of the Model will be the same as the Model it is derived from. This name can be edited if desired. It is handy to note in the Name the Model Number it is derived from. In this case the Model being used as the basis for the New Model is Model 601. The number for the Model is a new number that is arbitrarily assigned by the system. All new Models derived from Generic Models will start at 50,000 and count up.
5. Select the **Model Properties** tab. The default Fields available are listed. If more Tag values are required to be retrieved into the Stored Procedure then these, additional parameters can be added by selecting the **New** button.
6. Close this window to complete building the new Model Template.

**Assign New Model and the Specific Model Property Values**

1. Click on the Production Event from the upper window called **Events Enabled On This Unit**.
2. In the Available Models to Assign window, find the Model Number 50654 and hit the **Assign Model** button.
3. Specify each required Model Property for the newly assigned Model in the lower right window **Selected Model Properties**. Select each Model Property in the and select **Edit**.
4. A dialog appropriate for the Data Type of each Property will appear.
5. Select an appropriate item from the list presented, or type in an appropriate Value and select "OK".
6. To Activate the selected Model, hit the **Activate** button. Models with required Properties blank cannot be activated.
7. Reload the Event Manager.

**Generic Model 601-Archive Value Triggered**

Generic Models 600, 601, 602, and 603 are used to trigger the calling of a custom SQL Stored Procedure. The custom Stored Procedure allows the flexibility to use Result Sets to create, update, and delete data in the Plant Applications database. See the section on Result Sets for further explanation of their features.

**Model 601 Description**

When the historian tag value selected for the trigger tag changes, the model calls the defined stored procedure. The stored procedure has specific parameters that must be defined. The model passes data into those parameters and receives a message indicating success or failure. Historian values can also be passed into the stored procedure. The sampling type is a numeric value and can be found in the Sampling\_Type table in the Plant Applications SQL database. Result Set queries are created in SQL and the Event Manager Service handles these Result Sets as real time messages in Plant Applications. For information on result sets, see [Result Sets](#).

**Model 601 Properties**

The following Model properties are set up using the Plant Applications Administrator:

| Property                   | Description  |
|----------------------------|--|
| Maximum Run Time (Seconds) | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group     | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TriggerTag                 | Tag used to trigger the calling of the Stored Procedure.   |

## Models

|                                |  |
|--------------------------------|--|
| Local SP Name                  | Stored Procedure called when the model is triggered.   |
| Tag#1 (Optional)               | Tag#1 used to pass values into stored procedure.   |
| Tag#1 Sampling Type (Optional) | The Sampling Type used to acquire a value for Tag#1. The Sampling Type is a numeric value and can be found in the Sampling_Type table. |
| Tag#.....(Optional)            | Values can be retrieved for up to 50 Tags (Optional)   |

### Model 601 Stored Procedure Parameters

The following parameters are required at the beginning of the Stored Procedure that is called by Model 601. The stored procedure must have the same input and output parameters as defined and the parameter variables can be named whatever is desired but the order must be maintained. Up to 100 Historian Tag values can be passed into the Stored Procedure.

| SQL Variable Name                   | Parameter Description   |
|-------------------------------------|---|
| @ReturnStatus int OUTPUT,           | Flag to indicate Success or Failure (1-Success,0-Failure)   |
| @ReturnMessage varchar(255) OUTPUT, | Error Message to Write to Log File  |
| @EC_Id int,                         | Input to the Stored Procedure indicating the EC_Id of the model from the Event_Configuration table. |
| @TriggerTimestamp datetime,         | The New Time of the Triggering Tag  |
| @Hist1Value varchar(30),            | The Value For Tag 1 (Optional)  |
| @Hist2Value varchar(30),            | The Value For Tag 2 (Optional)  |
| @Hist3Value varchar(30),            | The Value For Tag 3 (Optional)  |
|                                     | Optional values can be retrieved for up to 50 Tags  |

To control how far back in time the Event Manager service looks for tag changes after a reload or restart of the Event Manager service, you can use Result Set 51 in your stored procedure. In the result set, use 'JumpToTime' for the ParameterName and for the ParameterValue, use the timestamp for how far back in time you want to look for tag changes. For more information, see the topic Result Set 51 in the online help.

### Sample Code

```

/***** Object:  Stored Procedure dbo.spLocal_PM1ScalEvents      Script Date:
8/29/00 12:18:39 PM *****/

if exists (select * from sysobjects where id =
object_id(N'[dbo].[spLocal_Model601]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)

drop procedure [dbo].[spLocal_Model601]

GO

SET QUOTED_IDENTIFIER OFF      SET ANSI_NULLS ON

GO

/***** Object:  Stored Procedure dbo.spLocal_Model601      Script Date:
8/29/00 12:19:00 PM *****/

CREATE PROCEDURE spLocal_Model601
@ReturnStatus int OUTPUT,
@ReturnMessage varchar(255) OUTPUT,
@EC_Id int,
@TimeStamp varchar(30),
@Lot_Num varchar(30),
@Unit_Num varchar(30)
AS

```

```

--Insert into local_info (field1,field2,field3) values
(convert (varchar(30),@EC_Id),convert (varchar(30),@TimeStamp),convert (varcha
r(30),@Lot_Num))
-- Return
Declare
@MaxEventTime datetime,
@Event_Num varchar(30),
@UnitInfo varchar(30),
@PU_Id int,
@EUIId int,
@EUTransaction_Type int,
@EUEvent_Id int,
@EUApplied_Product int,
@EUSource_Event int,
@EUEvent_Status int,
@EUConfirmed int,
@EUUser_Id int,
@EUPostUpdate int
-- Initialize variables.
Select @ReturnStatus = 1
Select @ReturnMessage = ''
Select @PU_Id = NULL
Select @PU_Id = PU_Id, @UnitInfo = Extended_Info From Event_Configuration
Where EC_Id = @EC_Id
If (@PU_Id Is NULL)
Begin
    Select @ReturnMessage = 'ECId=' + Convert(VarChar(5), @EC_Id) + '-PU_Id
is not defined in Event_Configuration record - ' + Convert(varchar(5),
@EC_Id)
    Goto Errorc
End
/* // Events Result Set
// -----
// 0 - Result Set Type (1)
// 1 - Id (Used for ordering within the Stored Procedure)
// 2 - Transaction_Type
// 3 - Event_Id
// 4 - Event_Num
// 5 - PU_Id
// 6 - TimeStamp
// 7 - Applied_Product
// 8 - Source_Event
// 9 - Event_Status
// 10 - Confirmed
// 11 - User_Id (Added Later, May not be there)

```

## Models

```
// 12 - PostUpdate (Added Later, May not be there)
*/
--Transaction_Type 1=Adds, 2=Updates, 3=Deletes
CREATE TABLE #EventUpdates (
    EUIId int,
    EUTransaction_Type int,
    EUEvent_Id int NULL,
    EUEvent_Num Varchar(25),
    EUPU_Id int,
    EUTimeStamp varchar(25),
    EUApplied_Product int Null,
    EUSource_Event int Null,
    EUEvent_Status int Null,
    EUConfirmed int Null,
    EUUser_Id int,
    EUPostUpdate int
)
Select @EUTransaction_Type = 1
--Select @EUEvent_Id = 0
Select @EUApplied_Product = NULL
Select @EUEvent_Status = 5
Select @EUConfirmed = 1
Select @EUUser_Id = 6
Select @EUPostUpdate = 0
/**
Hot Add
Execute spServer_DBMgrUpdEvent
@EUIId OUTPUT,
@NewEvent_Num, -- Event Num
@PU_Id, -- PU_Id
@EventTimestamp, -- Timestamp
@EUApplied_Product, -- Applied Product
@EUSource_Event, -- Source Id
@EUEvent_Status, -- Event Status
@EUTransaction_Type, -- @Transaction_Type int, -- 8: Input
0, -- @TransNum int, -- NewParam
@EUUser_Id, -- @UserId int, -- NewParam
Null, -- @CommentId int, -- NewParam
Null, -- @EventSubtypeId int, -- NewParam
Null, -- @TestingStatus int, -- NewParam
Null, -- @PropStartTime datetime, -- NewParam
Null, -- @PropEntryOn datetime, -- NewParam
0 -- @ReturnResultSet int -- NewParam
--Send insert to bus (post = 1, Pre = 0)
```

```

**/
Select @Event_Num = @UnitInfo + ltrim(rtrim(@Lot_Num)) +
ltrim(rtrim(@Unit_Num))
--Select @NewEvent_Num =
replace(replace(replace(right(convert(varchar(25),@Timestamp, 120),14), '
',''), '-',''), ':','')
Insert into #EventUpdates
(EUId,EUTransaction_Type,EUEvent_Id,EUEvent_Num,EUPU_Id,EUTimeStamp,EUAppli
ed_Product,EUSource_Event,EUEvent_Status,EUConfirmed,EUUser_Id,EUPostUpdate
)
Values(@EUId,@EUTransaction_Type,@EUId,@Event_Num,@PU_Id,Convert(VarChar(
30), @Timestamp,
120),@EUApplied_Product,NULL,@EUEvent_Status,@EUConfirmed,@EUUser_Id,@EUPos
tUpdate)
If (Select Count(EUPU_Id) From #EventUpdates) > 0
Begin
    Select ResultType = 1, *
    From #EventUpdates
    Order By EUId
End
Drop Table #EventUpdates
Select @ReturnStatus = 1
Select @ReturnMessage = 'Event Created for: ' + @Event_Num
Return
Errorrc:
    Select @ReturnStatus = 0
    Return
GO
GRANT EXECUTE ON dbo.spLocal_Model601 TO ComXClient
SET QUOTED_IDENTIFIER OFF SET ANSI_NULLS ON
GO

```

## Generic Model 602-Interval Triggered

Generic Models 600, 601, 602, and 603 are used to trigger the calling of a custom SQL Stored Procedure. The custom Stored Procedure allows the flexibility to use Result Sets to create, update, and delete data in the Plant Applications database. See the section on Result Sets for further explanation of their features.

### Model 602 Description

Model 602 is triggered by the timing interval defined in the model parameters. When the interval elapses, the Model calls the defined Stored Procedure. The stored procedure has specific parameters that must be defined. The model passes data into those parameters and receives a message indicating success or failure. Result Set queries are created in SQL and the Event Manager Service handles these Result Sets as real time messages in Plant Applications. Result Sets are defined further in another section.

### Model 602 Properties

The following Model properties are set up using the Plant Applications Administrator:



| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | The timing interval used to trigger the calling of the Stored Procedure.   |
| Delay(Seconds)                    | Type the number of seconds to add to the timestamp. Typically, this is used to adjust for data that falls outside the sampling window because Plant Applications cannot recognize millisecond resolution.  |
| Reserved                          |  |
| Reserved                          |  |
| Reserved                          |  |
| Local SP Name                     | Stored Procedure called when the model is triggered.   |

### Model 602 Stored Procedure Parameters

The following parameters are required at the beginning of the Stored Procedure that is called by Model 602. The stored procedure must have the same input and output parameters as defined and the parameter variables can be named whatever is desired but the order must be maintained. Up to 50 Historian Tag values can be passed into the Stored Procedure.

| SQL Variable Name                   | Parameter Description   |
|-------------------------------------|---|
| @ReturnStatus int OUTPUT,           | Flag to indicate Success or Failure (1-Success,0-Failure)   |
| @ReturnMessage varchar(255) OUTPUT, | Error Message to Write to Log File  |
| @EC_Id int,                         | Input to the Stored Procedure indicating the EC_Id of the model from the Event_Configuration table. |

### Sample Code

```

if exists (select * from sysobjects where id =
object_id(N'[dbo].[spLocal_TomCalcDependRedo]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[spLocal_TomCalcDependRedo]
GO
SET QUOTED_IDENTIFIER OFF      SET ANSI_NULLS ON
GO
create procedure dbo.spLocal_TomCalcDependRedo
@status int output,
@errmsg varchar(255) output,
@ecid int
as
declare
    @var_id int
select @status = 1
select @errmsg = ''
select @var_id = null
select @var_id = min(var_id) from tomtest

```

```

if (@var_id) is null
    return
select 2,@var_id,0,entry_by,0,result,convert(varchar(30),result_on),1,1
from tests where (var_id = @var_id) and (result_on between '10/27/00 10:00'
and '10/27/00 16:00') and (result is not null) and (canceled = 0)
select @var_id = null
select @var_id = min(var_id) from tomtest
if (@var_id) is null
    return
select 2,@var_id,0,entry_by,0,result,convert(varchar(30),result_on),1,1
from tests where (var_id = @var_id) and (result_on between '10/27/00 10:00'
and '10/27/00 16:00') and (result is not null) and (canceled = 0)
select @var_id = null
select @var_id = min(var_id) from tomtest
if (@var_id) is null
    return
select 2,@var_id,0,entry_by,0,result,convert(varchar(30),result_on),1,1
from tests where (var_id = @var_id) and (result_on between '10/27/00 10:00'
and '10/27/00 16:00') and (result is not null) and (canceled = 0)
select @var_id = null
select @var_id = min(var_id) from tomtest
if (@var_id) is null
    return
select 2,@var_id,0,entry_by,0,result,convert(varchar(30),result_on),1,1
from tests where (var_id = @var_id) and (result_on between '10/27/00 10:00'
and '10/27/00 16:00') and (result is not null) and (canceled = 0)
select @var_id = null
select @var_id = min(var_id) from tomtest
if (@var_id) is null
    return
select 2,@var_id,0,entry_by,0,result,convert(varchar(30),result_on),1,1
from tests where (var_id = @var_id) and (result_on between '10/27/00 10:00'
and '10/27/00 16:00') and (result is not null) and (canceled = 0)
select @var_id = null
select @var_id = min(var_id) from tomtest
if (@var_id) is null
    return
select 2,@var_id,0,entry_by,0,result,convert(varchar(30),result_on),1,1

```

## Models

```
from tests where (var_id = @var_id) and (result_on between '10/27/00 10:00'
and '10/27/00 16:00') and (result is not null) and (canceled = 0)
select @var_id = null
select @var_id = min(var_id) from tomtest
if (@var_id) is null
    return
select 2,@var_id,0,entry_by,0,result,convert(varchar(30),result_on),1,1
from tests where (var_id = @var_id) and (result_on between '10/27/00 10:00'
and '10/27/00 16:00') and (result is not null) and (canceled = 0)
select @var_id = null
select @var_id = min(var_id) from tomtest
if (@var_id) is null
    return
select 2,@var_id,0,entry_by,0,result,convert(varchar(30),result_on),1,1
from tests where (var_id = @var_id) and (result_on between '10/27/00 10:00'
and '10/27/00 16:00') and (result is not null) and (canceled = 0)
select @var_id = null
select @var_id = min(var_id) from tomtest
if (@var_id) is null
    return
select 2,@var_id,0,entry_by,0,result,convert(varchar(30),result_on),1,1
from tests where (var_id = @var_id) and (result_on between '10/27/00 10:00'
and '10/27/00 16:00') and (result is not null) and (canceled = 0)
delete from tomtest where var_id = @var_id
GO
SET QUOTED_IDENTIFIER OFF SET ANSI_NULLS ON
GO
GRANT EXECUTE ON [dbo].[spLocal_TomCalcDependRedo] TO [ComXClient]
GO
```

## Generic Model 603-Multiple Tags, Archive Value

### Triggered

#### Model 603 Description

Model 603 is triggered in two different ways depending on the setting of the Value Change Only property. If the Value Change Only is **false** (this is the default) then Model 603 is triggered by new historian tag archive values for any of the tags defined. When a new historian tag value is available in the archive for any tag, the model calls the defined stored procedure. The value does not have to change for the model to be triggered.

If the Value Change Only is **true** then Model 603 is triggered only by a change to consecutive historian tag archive values for any of the tags defined. When a value change is detected, the model calls the defined stored procedure. The value has to change for the model to be triggered.

The stored procedure has specific parameters that must be defined. The model passes data into those parameters and receives a message indicating success or failure. Only previous values and new values and their timestamps are passed to the stored procedure by Model 603. Historian values

can also be passed into the stored procedure. Result set queries are created in SQL and the Event Manager service handles these result sets as real-time messages in Plant Applications. For more information on result sets, see [Result Sets](#).

### Model 603 Properties

The following model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Local SP Name                     | Stored Procedure called when the model is triggered.   |
| Value Change Only                 | True or False (stored as 1 or 0)   |
| Delay(Seconds)                    | Type the number of seconds to add to the timestamp. Typically, this is used to adjust for data that falls outside the sampling window because Plant Applications cannot recognize millisecond resolution.  |
| Reserved                          |  |
| Tag#1 (Optional)                  | Tag#1 used to pass values into stored procedure.   |
| Tag# ... (Optional)               | Values can be retrieved for up to 96 Tags (Optional)   |

### Model 603 Stored Procedure Parameters

The following parameters are required at the beginning of the stored procedure that is called by Model 603. The parameter variables can be named whatever is desired, but the order must be maintained. If a string is passed in from an historian tag there is a 255 character limit on this string.

| SQL Variable Name              | Parameter Description  |
|--------------------------------|--|
| @ReturnStatus int<br>OUTPUT    | Flag to indicate Success or Failure (1-Success, 0-Failure)   |
| @ReturnMessage<br>varchar(255) | OUTPUT Error Message to Write to Log File  |
| @JumpToTime Datetime<br>OUTPUT | This is used to allow you to control how far back the EventManager goes in time to look for tag changes after a reload or restart of the Event Manager. By default the Event Manager goes back 3 days and looks for tags changes. In your code you could find the latest event on your unit and set the JumpToTime to be a second after this and then have it start looking from there for new events. |
| @EC_ID int                     | Input to the Stored Procedure indicating the EC_Id of the model from the Event_Configuration table.  |
| @ReservedInput1<br>varchar(30) | Reserved for future use.   |
| @ReservedInput2<br>varchar(30) | Reserved for future use.   |
| @ReservedInput3<br>varchar(30) | Reserved for future use.   |
| @ChangedTagNum int             | Indicates which tag of those configured for this model triggered the stored procedure to be called. The number is determined by the position of the tag in the Event Configuration wizard and ranges from 1 to n, n being the total number of tags configured for  |

## Models

|   |   |
|---|---|
|   | this model. For example for a model that has 10 tags configured, if the 8th tag changed causing this model to fire and this value would be set to 8. If multiple tags changed, this will contain the number of the first one in order entered in the event configuration. |
| @ChangedPrevValue<br>varchar(10)  | The previous value of the triggering tag. Again, the first one based on the order in the configuration if multiple tags fired.  |
| @ChangedNewValue<br>varchar(10)   | The new value of the triggering tag.  |
| @ChangedPrevTime<br>datetime  | The timestamp of the last value for this tag.   |
| @ChangedNewTime<br>datetime   | The timestamp of the triggering value for this tag.   |
| Note that the following variables will be filled in the same order that the tags are listed in the model configuration. In addition, all tag values and timestamps are true archive points not interpolated values. |   |
| @Tag1PrevValue<br>varchar(30)   | The previous value of the first tag.  |
| @Tag1NewValue<br>varchar(30)  | The current value the first tag.  |
| @Tag1PrevTime<br>datetime   | The previous timestamp of the first tag.  |
| @Tag1NewTime<br>datetime  | The current timestamp the first tag.  |
| @Tag2PrevValue<br>varchar(30)   | The previous value of the second tag.   |
| @Tag2NewValue<br>varchar(30)  | The current value the second tag.   |
| @Tag2PrevTime<br>datetime   | The previous timestamp of the second tag.   |
| @Tag2NewTime<br>datetime  | The current timestamp the second tag.   |
| @TagnPrevValue<br>varchar(30)   | The previous value of the nth tag.  |
| @TagnNewValue<br>varchar(30)  | The current value the nth tag.  |
| @TagnPrevTime<br>datetime   | The previous timestamp of the nth tag.  |
| @TagnNewTime<br>datetime  | The current timestamp the nth tag.  |

### Model 603 Stored Procedure Header

Here's a simple example of a model 603 stored procedure that has 2 tags in the event configuration of this model.

```
if exists (select * from sysobjects where id =
object_id(N'[dbo].[spLocal_Test603]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[spLocal_Test603]
GO
SET QUOTED_IDENTIFIER ON SET ANSI_NULLS ON
GO
CREATE PROCEDURE [dbo].[spLocal_Test603]
```

```

@ReturnStatus int OUTPUT,
@ReturnMessage varchar(255) OUTPUT,
@JumptoTime datetime OUTPUT,
@EC_Id int,
@Reserved1 varchar(30),
@Reserved2 varchar(30),
@Reserved3 varchar(30),
@ChangedTagNum int,
@ChangedPrevValue varchar(30),
@ChangedNewValue varchar(30),
@ChangedPrevTime datetime,
@ChangedNewTime datetime,
@Tag1PrevValue varchar(30),
@Tag1NewValue varchar(30),
@Tag1PrevTime datetime,
@Tag1NewTime datetime,
@Tag2PrevValue varchar(30),
@Tag2NewValue varchar(30),
@Tag2PrevTime datetime,
@Tag2NewTime datetime
AS
/*
--Use this to debug your stored procedure.
Insert into local_debug (time,msg)
  values (getdate(),(convert(varchar(10),@ChangedTagNum) + ' ' +
convert(varchar(30),@ChangedNewValue) + ' ' +
convert(varchar(30),@Tag1NewValue) + ' ' +
convert(varchar(30),@Tag2NewValue)))
*/
-- Initialize variables.
Select @ReturnStatus = 0 --Failure
Select @ReturnMessage = '' --Return a failure message here
/*
--Interrogate the incoming tag values and decide what to do.
-- You can use result sets to send messages to create or
-- update events or any other message type.
-- For example:
If @ChangedNewValue = '3.0' and @ChangedTagNum = 1
  Begin
    ...
  end
*/
-- Set output values appropriately.
Select @ReturnStatus = 1 --Success
Select @ReturnMessage = ''

```

## Models

Return

GO

```
SET QUOTED_IDENTIFIER OFF SET ANSI_NULLS ON
```

GO

```
/*
```

```
--Make sure to do this so the stored procedure will run!
```

```
*/
```

```
GRANT EXECUTE ON [dbo].[spLocal_Test603] TO [comxclient]
```

GO

## Generic Models 604-607

Generic Models 604, 605, and 607 are a variety of ODBC interface Models allowing various methods to interface to external systems using ODBC.

### Model 604-ODBC Data Import

Plant Applications must frequently exchange data with ODBC (Open Database Connectivity) compliant data sources such as ERP or legacy systems. A Plant Applications ODBC interface (Models 604, 605 or 607) can be used to read records from an ODBC data source and perform insertions, deletions, and updates with retries (Model 605) to an ODBC data source. It is assumed that the ODBC drivers for the target database have already been purchased, installed, configured, and tested by the customer. It is also required that a username and password are available with sufficient access rights to the remote database, and that a system DSN (Data Source Name) has been established on the Plant Applications server (Control Panel > Administrative tools > Data Sources > System DSN tab).

---

**IMPORTANT:** *The ODB connection must NOT have the parameter "Enable Lazy Close Support" selected to avoid locking the Plant Applications tables.*

---

### Configuration

General configuration and specific configuration for each type of data transfer for the Plant Applications-ODBC Interfaces will be performed in Plant Applications models and on variables in the Plant Applications Server using the Plant Applications Administrator. (A Plant Applications model performs tasks according to configurable model attributes in order to accomplish a set of defined tasks.)

### Startup and Shutdown

The Plant Applications File Interfaces run as a component or model inside of the Plant Applications Event Manager service. In order to start or stop the Plant Applications File Interfaces, the various models are activated or deactivated through the Plant Applications Administrator Configure Events option.

### Communication Protocol

The protocol of communication between the Plant Applications Server and other computers will be file transfer using FTP over TCP/IP. It can be assumed that both the Proficy Server and other computers will have appropriately configured IP addresses and will have FTP services loaded, tested, and fully functional. The actual transfer of the files can be done using the Plant Applications FTP Engine. This is configured using the Plant Applications Administrator. The testing of sending, retrieval, and deletion of files (Gets) using an FTP client is important to verify by using a program such as WS\_FTP prior to commissioning this interface. The Plant Applications FTP Engine is an FTP Client.

**Log Files**

All messages logged by the Plant Applications File Interfaces are logged in the Plant Applications Event Manager logfiles in the Plant Applications\Logfiles directory named EventMgr-01.Log with the 01 being the version of the log file. In the "r;Log" file there will be error messages and in the "r;Show" file EventMgr.Shw there is a summary of the interface's current configuration. By default the interface will log only error messages to the Log file including the date and time along with transaction specific data for the error. The Show file will contain details of the current configuration of the interface including which models are activated and the model parameters.

**Model 604 Description**

When Plant Applications is importing data from an ODBC data source, such as customer information from an ERP package, Model 604 is used to trigger the reading of that data on a time interval basis. The model calls a stored procedure, which then sends the passed in select query through the ODBC interface, processes the returned records, and transfers them to the Plant Applications database.

**Model 604 Properties**

The following Model 604 properties are set up using the Plant Applications Administrator.

| Name                              | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Min)               | Timing interval between calls  |
| ODBC Connect String               | For 32-bit SQL Server:<br>DSN=xxx;UID=yyy;PWD=zzz;SERVER=sss<br><br>For 64-bit SQL Server:<br>Driver={SQL Server};<br>Server=sss;Database=ddd;UID=yyy;PWD=zzz<br><br>For example:<br>Driver={SQL Server};<br>Server=USGBO;Database=GBDB;UID=sa;PWD=*****   |
| Select Statement                  | ODBC select statement to pass to the SP  |
| Local spName                      | The name of the SP that will import the data   |
| Update Statement                  | If doing status updating in SP, use a ? here   |
| Max Records                       | Number of records to bring back at one time  |

The ODBC Connect String property consists of the DSN name that was used to configure the ODBC interface, a valid User ID having select permissions on the needed tables, and the user's Password.

The Update Statement property can contain the update statement to be executed when the SP has successfully processed a new record set, or it can contain a partial update statement containing a question mark, which will be replaced by model 604's return text. If the update statement is entirely contained within the SP, a lone "?" must be used.

The Max Records property functions to limit the size of the data set returned by the select query. For example, if the select statement would normally return 117 customers, setting Max Records to 10 will result in processing those customers 10 at a time, and then updating a status field on the target table,



## Models

such as a last\_processed field or a STATUS field, to indicate that the record had been read by Plant Applications.

## Model 605-Generic ODBC Data Export

Plant Applications must frequently exchange data with ODBC (Open Database Connectivity) compliant data sources such as ERP or legacy systems. A Plant Applications ODBC interface (Models 604 or 607) can be used to read records from an ODBC data source and perform insertion, deletion, and update with retry (Model 605) to an ODBC datasource. It is assumed that the ODBC drivers for the target database have already been purchased, installed, configured, and tested by the customer. It is also required that a username and password are available with sufficient access rights to the remote database, and that a system DSN (Data Source Name) has been established on the Proficy Server (Control Panel | Administrative tools | Data Sources | System DSN tab).

---

*IMPORTANT: The ODB connection must NOT have the parameter "Enable Lazy Close Support" selected to avoid locking the Plant Applications tables.*

---

### Configuration

General configuration and specific configuration for each type of data transfer for the Plant Applications-ODBC Interfaces will be performed in Plant Applications models and on variables in the Plant Applications Server using the Plant Applications Administrator. (A Plant Applications model performs tasks according to configurable model attributes in order to accomplish a set of defined tasks.)

### Startup and Shutdown

The Plant Applications File Interfaces run as a component or model inside of the Plant Applications Event Manager service. In order to start or stop the Plant Applications File Interfaces, the various models are activated or deactivated through the Plant Applications Administrator Configure Events option.

### Communication Protocol

The protocol of communication between the Plant Applications Server and other computers will be file transfer using FTP over TCP/IP. It can be assumed that both the Proficy Server and other computers will have appropriately configured IP addresses and will have FTP services loaded, tested, and fully functional. The actual transfer of the files can be done using the Plant Applications FTP Engine. This is configured using the Plant Applications Administrator. The testing of sending, retrieval, and deletion of files (Gets) using an FTP client is important to verify by using a program such as WS\_FTP prior to commissioning this interface. The Plant Applications FTP Engine is an FTP Client.

### Log Files

All messages logged by the Plant Applications File Interfaces are logged in the Plant Applications Event Manager logfiles in the Proficy\Logfiles directory named EventMgr-01.Log with the 01 being the version of the log file. In the "r;Log" file there will be error messages and in the "r;Show" file EventMgr.Shw there is a summary of the interface's current configuration. By default the interface will log only error messages to the Log file including the date and time along with transaction specific data for the error. The Show file will contain details of the current configuration of the interface including which models are activated and the model parameters.

### Description

Model 605 supports insert, update, and delete queries to a remote database through an ODBC datasource and can be configured to require confirmation of the success of the query.

### Model 605 Properties

The following Model 605 properties are set up using the Plant Applications Administrator.

| Name                              | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Min)               | Timing interval between calls  |
| ODBC Connect String               | For 32-bit SQL Server:<br>DSN=xxx;UID=yyy;PWD=zzz;SERVER=sss<br><br>For 64-bit SQL Server:<br>Driver={SQL Server};<br>Server=sss;Database=ddd;UID=yyy;PWD=zzz<br><br>For example:<br>Driver={SQL Server};<br>Server=USGB0;Database=GBDB;UID=sa;PWD=*****   |
| Local spName                      | The name of the SP to run  |
| Confirm                           | 1=Confirm, 0=Do Not Confirm  |

The time interval parameter determines the wait time between calls to the Stored Procedure (SP) named in Parameter 3. The ODBC connect string names the Data Source Name (DSN) of the ODBC interface on the Proficy Server, a valid user ID and password with appropriate insert, update, and/or delete permissions in the needed tables, and for Oracle, the SERVER parameter names the Oracle instance name of the target database.

The SP will need to use a result set with the following fields:

**CheckExist:** The string to run against the foreign computer.

**Example:** "Select Count(\*) From GBX\_Rolls Where Roll\_Id='RPLG11B2201'"

**YesExist:** If the CheckExist call returned a value greater than 0, then the model would run this string against the foreign computer.

**Example:** "Update GBX\_Rolls Set RollTime='22-Feb-0118:45:33',  
RollOrder='453J389' Where Roll\_Id='RPLG11B2201'"

**NoExist:** If the CheckExist call returned a value of 0, then the model would run this string against the foreign computer.

**Example:** "Insert Into GBX\_Rolls (RollId, RollTime, RollOrder) Values  
( 'RPLG11B2201', '22-Feb-0118:45:33', '453J389' )"

**ConfirmString:** This is a confirmation string that would optionally (Parm4) be called against the foreign computer.

**Example:** "Select Count(\*) From GBX\_Rolls Where  
RollId='RPLG11B2201' And RollTime='22-Feb-0118:45:33' And RollOrder='453J389'"

**YesConfirm:** If the confirmation string returned a value greater

## Models

than 0, the model would call this string against the local database.

**Example:** "Update Local\_GBXRolls Set Complete=1 Where Id=45990"

**NoConfirm:** If the confirmation string returned a value of 0, the model would call this string against the local database.

**Example:** "Update Local\_GBXRolls Set Retry=Retry+1"

### Sample Stored Procedure

```
-----  
-----  
-- This SP is called by model 605 and is used to send a message to an ERP  
-- system when the production status for a production event is changed  
--  
-- Date          Version Build Author  Notes  
-- 17-Sep-2004  001      001    AlexJ dev  
-----  
-----  
ALTER PROCEDURE dbo.spLocal_ExportEventStatus  
@ReturnStatus Int OUTPUT,  
@ReturnMessage VarChar(255) OUTPUT  
AS  
-----  
-----  
-- Declare and set internal variables  
-----  
-----  
DECLARE @PurgeDelay Int,  
@RemoteQueueTable VarChar(50),  
@Now DateTime,  
@NowString VarChar(100),  
@ObjectName VarChar(255)  
SELECT @ObjectName = Object_Name(@@ProcId),  
@ReturnStatus = 1,  
@ReturnMessage = ''  
-----  
-----  
-- Extract parameters  
-----  
-----  
EXEC spLocal_CMNParameterLookup @PurgeDelay OUTPUT,  
@ObjectName, Null,1,'7'  
EXEC spLocal_CMNParameterLookup @RemoteQueueTable OUTPUT,  
@ObjectName, Null,2,'RemoteTable'  
SELECT @Now = GetDate()  
SELECT @NowString = '''' + Convert(Varchar(98), @Now, 121 ) + ''''  
-----  
-----
```

```

-- Delete records that have expired and were successful
-----
-----
DELETE Local_ExportEventStatus
WHERE Status = 'S'
AND DateDiff(dd, LastUpdated, @Now) >= @PurgeDelay
-----
-----

-- Create table to be returned to 605 model
-----
-----
CREATE TABLE #RemoteStatements
(
  Id Int IDENTITY (1, 1) NOT NULL ,
  CheckExist Varchar(1000) Null,
  YesExist Varchar(1000) Null,
  NoExist Varchar(1000) Null,
  ConfirmString Varchar(1000) Null,
  YesConfirm Varchar(1000) Null,
  NoConfirm Varchar(1000) Null
)
-----
-----

-- Populate table to be returned to 605 model
-----
-----
INSERT #RemoteStatements (CheckExist, YesExist, NoExist, ConfirmString,
YesConfirm, NoConfirm)
SELECT
'SELECT Count(LotId) FROM ' + @RemoteQueueTable
+ ' WHERE LotId = ''' + RTrim(LTrim(LotId)) + ''',
'SELECT 1',
'INSERT ' + @RemoteQueueTable + ' (LotId, LotStatus) VALUES ('
+ IsNull('''' + RTrim(LTrim(LotId)) + ''', 'Null') + ', '
+ IsNull('''' + RTrim(LTrim(LotStatus)) + ''', 'Null') + ')',
'SELECT Count(LotId) FROM ' + @RemoteQueueTable
+ ' WHERE LotId = ''' + RTrim(LTrim(LotId)) + ''',
'UPDATE Local_ExportEventStatus SET Status = 'S', LastUpdated = ' +
@NowString +
' Where Id = ' + Convert(VarChar(20), Id),
'UPDATE Local_ExportEventStatus SET Status = 'F', LastUpdated = ' +
@NowString +
' Where Id = ' + Convert(VarChar(20), Id)
FROM Local_ExportEventStatus
WHERE Status = 'N'
ORDER

```

## Models

BY Id

```
-----  
----  
-- Return 605 table  
-----  
----
```

```
SELECT CheckExist, YesExist, NoExist, ConfirmString, YesConfirm, NoConfirm  
FROM #RemoteStatements  
ORDER  
BY Id  
DROP TABLE #RemoteStatements  
RETURN  
GO  
SET QUOTED_IDENTIFIER OFF  
GO  
SET ANSI_NULLS ON  
GO  
GRANT EXECUTE ON [dbo].[spLocal_ExportEventStatus] TO [comxclient]  
GO
```

## Model 607-Generic ODBC Data Import

Plant Applications must frequently exchange data with ODBC (Open Database Connectivity) compliant data sources such as ERP or legacy systems. A Plant Applications ODBC interface can be used to read records from an ODBC data source and perform insertion, deletion, and update with retry (Model 605) to an ODBC datasource. It is assumed that the ODBC drivers for the target database have already been purchased, installed, configured, and tested by the customer. It is also required that a username and password are available with sufficient access rights to the remote database, and that a system DSN (Data Source Name) has been established on the Proficy Server (Control Panel | Administrative tools | Data Sources | System DSN tab).

---

**IMPORTANT:** *The ODB connection must NOT have the parameter "Enable Lazy Close Support" selected to avoid locking the Plant Applications tables.*

---

### Configuration

General configuration and specific configuration for each type of data transfer for the Plant Applications-ODBC Interfaces will be performed in Plant Applications models and on variables in the Plant Applications Server using the Plant Applications Administrator. (A Plant Applications model performs tasks according to configurable model attributes in order to accomplish a set of defined tasks.)

### Startup and Shutdown

The Plant Applications File Interfaces run as a component or model inside of the Plant Applications Event Manager service. In order to start or stop the Plant Applications File Interfaces, the various models are activated or deactivated through the Plant Applications Administrator Configure Events option.

### Communication Protocol

The protocol of communication between the Plant Applications Server and other computers will be file transfer using FTP over TCP/IP. It can be assumed that both the Proficy Server and other computers

will have appropriately configured IP addresses and will have FTP services loaded, tested, and fully functional. The actual transfer of the files can be done using the Plant Applications FTP Engine. This is configured using the Plant Applications Administrator. The testing of sending, retrieval, and deletion of files (Gets) using an FTP client is important to verify by using a program such as WS\_FTP prior to commissioning this interface. The Plant Applications FTP Engine is an FTP Client.

**Log Files**

All messages logged by the Plant Applications File Interfaces are logged in the Plant Applications Event Manager logfiles in the Plant Applications\Logfiles directory named EventMgr-01.Log with the 01 being the version of the log file. In the "r;Log" file there will be error messages and in the "r;Show" file EventMgr.Shw there is a summary of the interface's current configuration. By default the interface will log only error messages to the Log file including the date and time along with transaction specific data for the error. The Show file will contain details of the current configuration of the interface including which models are activated and the model parameters.

**Model 607 Properties**

The following Model 607 properties are set up using the Plant Applications Administrator:

| Name                              | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Min)               | Timing interval between calls  |
| Local spName                      | The name of the SP called each interval  |
| Debug Mode                        | 0=Normal Mode, 1= Debug Mode   |

**Description**

Every Interval (Property1), Model 607 will call the local stored procedure (Property2). The model will pass no parameters and expect no return codes or output parameters for this stored procedure. Model 607 code ONLY expects a single result set in the order you desire it to execute things. The result set that the model code is looking for has 1, 2, or 6 columns. If the result set has 1 or 2 columns this means that you are NOT giving Model 607 any more sql statements to run but are trying to indicate to the model that the previously executed sql statement failed or succeeded. In such a case, column 1 must be a value of 0 or 1, with 1 meaning success. Column 2 can be a string that the model will dump to the log file if column 1 was a 0. NO resultset or a resultset containing more than 2 columns will be interpreted as success for the previous statement. Every INTERVAL, the model will cache the Database connections for the wakeup duration and let them go at the end of the wakeup duration. What this means is that the model will connect to your remote DB(s) Interval (Property1), but only once per unique connection.

The 6-column result set looks like this:

- Original SQL Connect String
- Original SQL
- Failed SQL Connect String
- Failed SQL
- Success SQL Connect String
- Success SQL

## Models

Connect Strings are either nothing, meaning 'r;', or a valid ODBC connect string such as 'r;DSN=GBDS;UID=comxclient;PWD=comxclient;'. For an empty string the code assumes you want to run your sql against GBDB and will use the database connection already associated with the Event Mgr.

This process will continue on and on until no more of the sql statements the model has been asked to run return result sets with more than 2 columns.

The confusing thing with this model is that you are telling it success or failure via a result set plus giving the model instructions via a result set.

## Model 650-Event Conformance \ Test Percentage Complete

Model 650 is used to calculate conformance data for specified variables for specific events. It also calculates the Testing Percentage Complete data for each variable specified.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 650 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing interval the model fires.   |
| Variable Update Delay (seconds)   | This is the minimum number of seconds between recalculations. If the previous calculation was done less than this amount of time ago, then the calculation doesn't get fired. This must be set to a value greater than 30 seconds  |
| Cache Reload Window               | This is the number of seconds that the specification cache for a product will remain valid. After the specified number of seconds expires, the cache will get reloaded.  |

## Import Models

### Model 50

Model 50 is a file import interface model. It creates production events, grade changes, and updates the Run Number variable based on data from a specific file format from the Acquidata System.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 50 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |

|                           |   |
|---------------------------|---|
| Model Processing Group    | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information. |
| First Char Of File        | First character of the filename to look for in the Import File Location directory.          |
| Work Unit                 | Machine Number\Acquidata Work Unit.   |
| TINT:Interval (Minutes)   | Timing Interval the model checks for New Import Files.                                      |
| Import File Location Spec | Drive and directory for Import Files.   |
| Accepted File Path        | Drive and directory for Successfully Processed Files.                                       |
| Rejected File Path        | Drive and directory for Rejected Files.   |
| (Var Desc) Jumbo Number   |   |
| Purge Days                | # Days of Accepted/Rejected files to keep.  |
| Grade Changes (TRUE)      | Create Grade Changes from the file. Default is True.  |

### File Layout

| Field                            | Format                        | Description                                      |
|----------------------------------|-------------------------------|--|
| Transaction Type                 | X(1)                          | "A" For Add Test Transaction                     |
| Work Unit                        | X(2)                          | Machine Number\Acquidata Work Unit               |
| Test Code                        | X(10)                         | Acquidata Test Code                              |
| SID#1                            | X(12)                         | Grade Code                                       |
| SID#2                            | X(12)                         | Run Number                                       |
| SID#3                            | X(12)                         | Reel Number                                      |
| Acquisition Date                 | YYYYMMDD(8)                   | Entered Date                                     |
| Acquisition Time                 | HHMM(4)                       | Entered Time                                     |
| Average Of Readings              | Z9999.999(10)                 | Average Of Test                                  |
| Number Of Reel Position Averages | Z999(3)                       | Number Of Reel Position Averages                 |
| Readings                         | Z999(3) + Z9999.999(10) * (n) | Number Readings In Position AND Position Average |

## Model 51

Model 51 is a file import interface model.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 51 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Purge Days                        | # Days of Accepted/Rejected files to keep.   |

## Model 52



## Models

Model 52 is a file import interface model. It creates events from a file from a specific file format.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 52 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Month Chars                       | The Tappi Month Characters used to generate the M character.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |
| Purge Days                        | # Days of Accepted/Rejected files to keep.   |
| Tappi Begin                       | This value is used as the 1 <sup>st</sup> characters of the Tappi or Event Number.   |
| Grade Changes (TRUE)              | Create Grade Changes from the file. Default is True.   |

### Model 53

Model 53 is a file import interface model. It creates events and grade changes from a specific file format.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 53 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Tappi Begin                       | This value is used as the 1 <sup>st</sup> characters of the Tappi or Event Number.   |

|                      |   |
|----------------------|---|
| Prod Day (HH:MM)     | The time that the counter is restarted for the number of events made in that day. |
| Month Chars          | The Tappi Month Characters used to generate the M character.                      |
| Purge Days           | # Days of Accepted/Rejected files to keep.  |
| Grade Changes (TRUE) | Create Grade Changes from the file. The default is True.                          |
| Default Product Code |   |

## Model 54

Model 54 is the Plant Applications-PPR Interface. The purpose of the PPR reel interface is to automate and synchronize the reel turn-up events and reel quality data between PPR and Plant Applications. Additionally, some sites opt to include a release variable option and a specification transfer option.

The primary PPR reel interface is a uni-directional message from PPR to Plant Applications. Fixed port numbers are assigned to receive messages from PPR. The messages are dumped to a flat data file in a defined folder location for processing by Plant Applications input model 54. The format of the content of the message from PPR is fixed for each mill. The PPR message contains both basic reel data and the scanner sensor data. Plant Applications variables are configured to connect to the PPR message content through the syntax of the InputTag references.

---

*To access the support site, you'll need a user name and password. To get an account, contact us at [ProficiencySupport@GE.com](mailto:ProficiencySupport@GE.com).*

---

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 54 Properties

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT: Interval (Minutes )         | The frequency at which the import file location is checked to find new files.  |
| Import File Location              | This is the location that the PPR files are sent before being processed. The site specific paths must match the statements in the PPR-Receive.reg, PPR-Receive-CreateService.bat and the actual folder locations that were created.  |
| Accepted File Path                | This is the path where properly processed files are moved. The site specific paths must match the statements in the PPR-Receive.reg, PPR-Receive-CreateService.bat and the actual folder locations that were created.  |
| Rejected File Path                | This is the path where improperly processed files are moved. The site specific paths must match the statements in the PPR-Receive.reg, PPR-Receive-CreateService.bat and the actual folder locations that were created.  |
| Machine Code                      | This will differ for each site and may differ from the PM number. It should match the first two digits of the "sample id" sent in the *.dat file.  |
| Grade Changes (TRUE)              | Determines whether PPR will send grade change messages in addition to reel quality data messages.  |
| Purge Days                        | *.dat files remain in the PPRInterface folders for the selected number of days.  |

## Models

|                                |   |
|--------------------------------|---|
| Prod Day (HH:MM)               | The time at which the reel number (event number) date changes and the sequence digits are reset to zero. See Timestamp discussion below.  |
| Tappi Begin                    | Enter a string value if fixed characters are to be prefixed to Tappi.   |
| Value Index to take onto Tappi | Enter any PPR buffer index of a variable to append to the Tappi code if desired.  |
| Version                        | Rarely used. If equal to 2, then event numbers will not be prefixed with the 2-digit year code and code for event number uniqueness is added.   |
| Timestamp Index                | This is the PPR buffer index reference for the timestamp. If an index position is entered, Plant Applications uses the PPR date found at this offset to create the event. A PPR value at index 15 has the format YYYYMMDD. If blank, Plant Applications will determine the date and reel count with respect to the Prod Day entry. See following paragraph. |

### Model 55

Model 55 is a file import interface model. It creates events and grade changes from a specific file format.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

#### Model 55 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Grade Changes (TRUE)              | Create Grade Changes from the file. The default is True.   |

### Model 56

Model 56 is a test data import interface model. It imports test data from a specific file format.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

#### Model 56 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |

|                           |   |
|---------------------------|---|
| Model Processing Group    | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information. |
| TINT:Interval (Minutes)   | Timing Interval the model checks for New Import Files.                                      |
| Import File Location Spec | Drive and directory for Import Files.   |
| Accepted File Path        | Drive and directory for Successfully Processed Files.                                       |
| Rejected File Path        | Drive and directory for Rejected Files.   |

## Model 57

Model 57 is a test data import interface model. It imports time based test data from a specific file format.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 57 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |

## Model 58

Model 58 is a test data import interface model. It imports time based test data from a specific file format.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 58 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Purge Days                        | # Days of Accepted/Rejected files to keep.   |
| Time stamp (Ex. 07:00:00)         | Format of the timestamp.   |

## Models

|  |   |
|--|---|
| Search String For Date Line (Ex. MILWAUKEE METROPOLITAN) | Search String indicating the line that includes the date. |
|--|---|

### Model 59

Model 59 is a test data import interface model. It imports time based test data from a specific file format.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

#### Model 59 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Purge Days                        | # Days of Accepted/Rejected files to keep.   |
| Time stamp (Ex. 07:00:00)         | Format of the timestamp.   |

### Model 64

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

Model 64 is a test data import interface model. It imports customer data from a specific file format.

Model 64 watches for incoming files, then it calls a user defined stored procedure for each line in the file. The line in the file is broken up into 23 parts:

```
Params[0] = TheLine.Mid(0,6);
Params[1] = TheLine.Mid(6,6);
Params[2] = TheLine.Mid(12,3);
Params[3] = TheLine.Mid(15,1);
Params[4] = TheLine.Mid(16,30);
Params[5] = TheLine.Mid(46,20);
Params[6] = TheLine.Mid(66,15);
Params[7] = TheLine.Mid(81,2);
Params[8] = TheLine.Mid(83,2);
Params[9] = TheLine.Mid(85,3);
Params[10] = TheLine.Mid(88,10);
Params[11] = TheLine.Mid(98,1);
Params[12] = TheLine.Mid(99,20);
Params[13] = TheLine.Mid(119,20);
```

```
Params[14] = TheLine.Mid(139,1);
Params[15] = TheLine.Mid(140,6);
Params[16] = TheLine.Mid(146,10);
Params[17] = TheLine.Mid(156,2);
Params[18] = TheLine.Mid(158,2);
Params[19] = TheLine.Mid(160,6);
Params[20] = TheLine.Mid(166,1);
Params[21] = TheLine.Mid(167,5);
Params[22] = TheLine.Mid(172,3);
```

After being converted to uppercase and removing all leading and trailing spaces, these 23 parts are passed to the stored procedure as varchar parameters.

### Model 64 Properties

The following model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing interval the model checks for new import files.   |
| Import File Location Spec         | Drive and directory for import files.  |
| Accepted File Path                | Drive and directory for successfully processed files.  |
| Rejected File Path                | Drive and directory for rejected files.  |
| Purge Days                        | # Days of accepted/rejected files to keep  |
| Local Save spName                 | Stored procedure name called to write the data to the database.  |

### Model 65

Model 65 is a test data import interface model. It imports consignee data from a specific file format.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 65 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Purge Days                        | # Days of Accepted/Rejected files to keep  |

|                   |   |
|-------------------|---|
| Local Save spName | Stored Procedure Name called to write the data to the database. |
|-------------------|---|

## Model 66

Model 66 is a test data import interface model. It imports roll data from a specific file format.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 66 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Purge Days                        | # Days of Accepted/Rejected files to keep  |
| Local Save spName                 | Stored Procedure Name called to write the data to the database.  |

## Model 67

Model 67 is a test data import interface model. It imports chip test data from a specific file format.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 67 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Purge Days                        | # Days of Accepted/Rejected files to keep  |

## Model 68

Model 68 is a test data import interface model. It imports test data from a specific file format.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 68 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Purge Days                        | # Days of Accepted/Rejected files to keep  |

### Model 69

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

Model 69 is a test data import interface model. It imports TSO test data from a specific file format.

#### Input File Format

Line 1: PU\_Id

Line 2: Event\_Num

Line 3 – Line 17: Not used

Line 18: Property Id

Line 19: Not Used

Line 20: Not Used

Line 21: #Values Val1 Val2 .. Valn (tab delimited data)

For Line 1, build a string, such as "PM=<Line1>", and cross-reference that to the Extended\_Info field to find the production unit. Then, verify that the Event\_Num in Line 2 exists on the production unit. If Event\_Num does not exist, you will get a warning message and the file will not be processed. Ensure that all the values are present in Line 21. For example, if the #Values = 3, then that value is incremented by 1 to ensure there are four tab-delimited values: one for the number of values plus the three values. To find the variable, create a cross-reference string (for example, TSO\<<property ID from line 18>\<value position 1-n>. The cross-reference string is generated for each of the n values in the line 21 data values. The cross-reference string is then used to see if a variable has been defined with the input\_tag. If a variable is found, then the corresponding value is sent into that variable at the time of the event.

### Model 69 Properties

The following model properties are set up using the Plant Applications Administrator:

| Property                          | Description   |
|-----------------------------------|---|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only |



## Models

|                           |  |
|---------------------------|--|
|                           | if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group    | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.      |
| TINT:Interval (Minutes)   | Timing interval the model checks for new import files.   |
| Import File Location Spec | Drive and directory for import files.  |
| Accepted File Path        | Drive and directory for successfully processed files.  |
| Rejected File Path        | Drive and directory for rejected files.  |
| Purge Days                | Number of days to keep accepted/rejected files.  |

## Model 70

Model 70 imports data from Promix after every interval is elapsed. The data is retrieved from an external database using an ODBC datasource.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

**IMPORTANT:** *The ODB connection must NOT have the parameter "Enable Lazy Close Support" selected to avoid locking the Plant Applications tables.*

---

### Model 70 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                                     | Description  |
|--|--|
| <b>Maximum Run Time (Seconds)</b>            | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group                       | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:(Interval) Minutes                      | A value change of this tag is used to trigger a new production event to be created.  |
| ODBC Connect String                          | DSN=xx;UID=yyy; PWD=zzz;   |
| Warehouse (Ex. LA)                           |  |
| GetFunc (Ex. GBK_Get_Customer_Order)         |  |
| ConfirmFunc (Ex. GBK_Confirm_Customer_Order) |  |
| Select1 (Ex. SOP_UNCONFIRMED_TRANSACTIONS)   |  |
| Select2 (Ex. SOP_CHAR_UNCONFIRMED_TRANS)     |  |
| Local Order spName                           |  |
| Local Order Char spName                      |  |
| CustAddrTable (Ex. Customer_Addresses)       |  |
| ??? (Ex. 03)                                 |  |
| ??? (Ex. 60)                                 |  |

## Model 71

Model 71 imports Roll Disposition data after every Interval is elapsed. The data is retrieved from an external database using an ODBC data source.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

**IMPORTANT:** The ODB connection must NOT have the parameter "Enable Lazy Close Support" selected to avoid locking the Plant Applications tables.

### Model 71 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                                       | Description  |
|--|--|
| <b>Maximum Run Time (Seconds)</b>              | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group                         | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:(Interval) Minutes                        | A value change of this tag is used to trigger a new production event to be created.  |
| ODBC Connect String                            | DSN=xx;UID=yyy; PWD=zzz;   |
| Warehouse (Ex. LA)                             |  |
| GetFunc (Ex. GBK_Get_Roll_Disposition)         |  |
| ConfirmFunc (Ex. GBK_Confirm_Roll_Disposition) |  |
| Select1 (Ex. IC_DISP_CHANGES_UNCONF_TRANS)     |  |
| InputTag (Ex. PDC=Disposition)                 |  |
| UserId for Var Update                          |  |

### Model 74

Model 74 exports QC data after every Interval is elapsed. The data is exported to an external database using an ODBC data source.

*The functionality in this model is available only if you have purchased the Quality Module.*

**IMPORTANT:** The ODB connection must NOT have the parameter "Enable Lazy Close Support" selected to avoid locking the Plant Applications tables.

### Model 74 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:(Interval) Minutes           | A value change of this tag is used to trigger a new production event to be created.  |

### Model 75: Fixed-width File Import

*The functionality in this model is available only if you have purchased the Quality Module.*

## Models

The Plant Applications File interface models allow for the export of structured or delimited data from Plant Applications into a file, and for the import of file data into Plant Applications. A Plant Applications calculation that is either time based or event based triggers the export of data, which is accomplished through calling a stored procedure (SP) from a Plant Applications calculation. The SP does much of the work via its select and update queries, which gather up the information, often into temporary tables, and then writes it out to a file using Plant Applications specific Result Sets to generate the file, its name, and its format. Imports are done by having files FTP'd to the Proficy Server and by configuring a specific model to watch a directory for a new file and specifying a stored procedure to be called by the model to handle the importing of the data into Plant Applications.

### Configuration

General configuration and specific configuration for each type of transfer for the Plant Applications File Interfaces will be performed in Plant Applications models and on variables in the Plant Applications Server using the Plant Applications Administrator. (A Plant Applications model performs tasks according to configurable model attributes in order to accomplish a set of defined tasks.)

### Startup and Shutdown

The Plant Applications File Interfaces run as a component or model inside of the Plant Applications Event Manager service. In order to start or stop the Plant Applications File Interfaces, the various models are activated or deactivated through the Plant Applications Administrator Configure Events option.

### Communication Protocol

The protocol of communication between the Plant Applications Server and other computers will be file transfer using FTP over TCP/IP. It can be assumed that both the Proficy Server and other computers will have appropriately configured IP addresses and will have FTP services loaded, tested, and fully functional. The actual transfer of the files can be done using the Plant Applications FTP Engine. This is configured using the Plant Applications Administrator. The testing of sending, retrieval, and deletion of files (Gets) using an FTP client is important to verify by using a program such as WS\_FTP prior to commissioning this interface. The Plant Applications FTP Engine is an FTP Client.

### Log Files

All messages logged by the Plant Applications File Interfaces are logged in the Plant Applications Event Manager log files in the Proficy\Logfiles directory named EventMgr-01.Log with the 01 being the version of the log file. In the "r;Log" file there will be error messages and in the "r;Show" file EventMgr.Shw there is a summary of the interface's current configuration. By default the interface will log only error messages to the Log file including the date and time along with transaction specific data for the error. The Show file will contain details of the current configuration of the interface including which models are activated and the model parameters.

### Model 75 Description

Model 75 will parse an input file and pass specific components of each line of the fixed width file to a stored procedure for processing. Every Interval a specific directory will be checked for new files. The user configures a series of desired start positions and field lengths. This model will parse the input file and pass one row of data at a time into the stored procedure named in spName. Each row will be passed into the stored procedure one row at a time. Each field (start positions and field lengths) will be imported into a parameter in the stored procedure starting with parameter #6. The first 5 parameters ReturnStatus, ReturnMessage, User\_Id, Event\_Config\_PU, and Event\_Config\_Model are required in the stored procedure for Model 75 to work correctly. They can be named different but must exist in that order.

Model 75 code automatically determines when the End of File (EOF) has been reached, so it is not necessary to include an EOF character in the stored procedure.

```
CREATE PROCEDURE dbo.SampleReelFileImport
```

@ReturnStatus int OUTPUT,  
 @ReturnMessage varchar(255) OUTPUT,  
 @User\_Id int,  
 @Event\_Config\_PU int,  
 @Event\_Config\_Model int,  
 @Field1 Varchar(25)

**Model 75 Properties**

Configure the following fields to set up the model in the Plant Applications Administrator.

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval Plant Applications checks for New Import Files   |
| Import File Location              | Drive and directory, extensions may vary   |
| Accepted File Path                | Drive and directory for Successfully Processed Files   |
| Rejected File Path                | Drive and directory for Rejected Files   |
| Purge Days                        | # Days of files to keep  |
| Local spName                      | Called for each row in the file  |
| Max Line Length (Bytes)           | Maximum Line Length to use if no end is found (Bytes)  |
| AddTimeToFile (TRUE)              | If TRUE date is added to filename<br>yyymmddhhmiss   |
| Start Pos,Len                     | e.g. 1,10 (start at position 1 for a length of 10 characters)  |
| Start Pos,Len (optional)          | e.g. 11,8 (start at position 11 for a length of 8 characters)  |
| Start Pos,Len (optional)          | Add Pos,Len values for each field to be imported   |

*See the GE Support Site to download an example of a Model 75 Import Stored Procedure.*

**Model 76**

Model 76 imports data after every Interval is elapsed. The data is imported using customized stored procedures.

*The functionality in this model is available only if you have purchased the Quality Module.*

**Model 76 Properties**

The following Model properties are set up using the Plant Applications Administrator:

| Property | Description |
|----------|-------------|
|----------|-------------|

|                                   |  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:(Interval) Minutes           | A value change of this tag is used to trigger a new production event to be created.  |
| Local GetIds spName               |  |
| Local ProcessId spName            |  |
| Local ConfirmId spName            |  |

## Model 79 Delimited File Import

*The functionality in this model is available only if you have purchased the Quality Module.*

The Plant Applications File interface models allow for the export of structured or delimited data from Plant Applications into a file, and for the import of file data into Plant Applications. A Plant Applications calculation that is either time based or event based triggers the export of data, which is accomplished through calling a stored procedure (SP) from the calculation. The SP does much of the work via its select and update queries, which gather up the information, often into temporary tables, and then writes it out to a file using Plant Applications specific Result Sets to generate the file, its name, and its format. Imports are done by having files FTP'd to the Proficy Server and by configuring a specific model to watch a directory for a new file and specifying a stored procedure to be called by the model to handle the importing of the data into Plant Applications.

### Configuration

General configuration and specific configuration for each type of transfer for the Plant Applications File Interfaces will be performed in Plant Applications models and on variables in the Proficy Server using the Plant Applications Administrator. (A Plant Applications model performs tasks according to configurable model attributes in order to accomplish a set of defined tasks.)

### Startup and Shutdown

The Plant Applications File Interfaces run as a component or model inside of the Plant Applications Event Manager service. In order to start or stop the Plant Applications File Interfaces, the various models are activated or deactivated through the Plant Applications Administrator Configure Events option.

### Communication Protocol

The protocol of communication between the Proficy Server and other computers will be file transfer using FTP over TCP/IP. It can be assumed that both the Proficy Server and other computers will have appropriately configured IP addresses and will have FTP services loaded, tested, and fully functional. The actual transfer of the files can be done using the Plant Applications FTP Engine. This is configured using the Plant Applications Administrator. The testing of sending, retrieval, and deletion of files (Gets) using an FTP client is important to verify by using a program such as WS\_FTP prior to commissioning this interface. The Plant Applications FTP Engine is an FTP Client.

### Log Files

All messages logged by the by the Plant Applications File Interfaces are logged in the Plant Applications Event Manager logfiles in the Plant Applications\Logfiles directory named EventMgr-01.Log with the 01 being the version of the log file. In the "Log" file there will be error messages and in the "Show" file EventMgr.Shw there is a summary of the interface's current configuration. By default the interface will log only error messages to the Log file including the date and time along with

transaction specific data for the error. The Show file will contain details of the current configuration of the interface including which models are activated and the model parameters.

**Model 79 Description**

Model 79 will parse input files and pass specific components of each line of the delimited file to a stored procedure for processing. Every Interval a specific directory will be checked for new files. Each configured field will be imported into a parameter in the stored procedure starting with parameter #5. The first 4 parameters ReturnStatus, ReturnMessage, User\_Id, and EC\_Id are required in the stored procedure for Model 79 to work correctly. They can be named different but must exist in that order.

```
CREATE PROCEDURE spLocal_ImportData
@ReturnStatus int OUTPUT,
@ReturnMessage varchar(255) OUTPUT,
@User_Id int,
@EC_Id int,
@Field1 varchar(25)
.
.
.
@Fieldn
```

The stored procedure that processes the lines of the import file needs to do end of file checking, as it will be called once per line, and will be passed the string value EOF into the first parameter when the EOF character is hit at the bottom of the text file. Appendix A contains spLocal\_DelimitedImport in its entirety as an example.

The @Field parameters match up to the columns selected in the configuration of the model and are described below as the Column properties. For each column you wish to reference you must have a corresponding @Field1 parameter. Again the name can be anything you wish within SQL standards. Except for the first @Field parameter as mentioned above, the type of the remaining parameters can be any supported SQL as long as it makes sense with the data in the file. For example if the data in the file is say a value of 64000 then the type for the corresponding @Field parameter cannot be a tinyint because this type has a limit of 256.

The @ReturnStatus parameter is used to indicate to the Event Manager that the stored procedure executed successfully. A value of 0 indicates it failed and a message will be written to the Event Manager log file. Any value greater than 0 indicates success.

**Model Configuration Properties**

Configure the following fields to set up the model in the Plant Applications Administrator.

| Property                   | Example Value | Description  |
|----------------------------|---------------|--|
| Maximum Run Time (Seconds) |               | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |

Models

|                          |  |   |
|--------------------------|--|---|
| Model Processing Group   | Up to six digits                           | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.   |
| TINT:Interval (Min)      | TINT:1                                     | Timing Interval Plant Applications checks for New Import Files  |
| Import File Location     | C:\Plant Applications\Files\Incoming\*.txt | Drive and directory, extensions may vary  |
| Accepted File Path       | C:\Plant Applications\Files\Processed      | Drive and directory for Successfully Processed Files  |
| Rejected File Path       | C:\Plant Applications\Files\Unprocessed    | Drive and directory for Rejected Files  |
| Purge Days               | 7  | # Days of Processed or Unprocessed files to keep  |
| Local spName             | spLocal_procedure_name                     | Called for each row in the file   |
| Max Line Length (Bytes)  | 1000                                       | Maximum Line Length to use if no end is found (Bytes)   |
| AddTimeToFile (TRUE)     | True                                       | TRUE adds -yyyymmddhhmiss to file name  |
| ASCII Code for Delimiter | 44   | For example, comma is ASCII 44  |
| Column #                 |  | The first column you want to reference. This will be the data passed in to the 5th parameter in the stored procedure named above. For example if there are 12 columns of data in the file each separated by a comma, you may want to ignore some of the columns. This property gives you that opportunity. At least one column must be referenced so the first instance of this property is required (note that at this time, the Event Configuration screen in the Plant Applications Administrator indicate this is an optional property). Important: This first column is used by the Event Manager to send a value of 'r;EOF' to signify that there are no more lines to process in the text file. Therefore this column must be of a type a char or varchar when referenced in the stored procedure even if the value is a number. |
| Column # (Optional)      |  | The next column you want to reference. This will be the data passed in to the 6th parameter in the stored procedure named above.  |
| Column # (Optional)      |  | The next column you want to reference, etc. This will be the data passed in to the nth  |

|  |  |  |
|--|--|--|
|  |  | parameter in the stored procedure named above. |
|--|--|--|

## Appendix A - Delimited File Import Example

The following SP is an example of how to create production events and update Plant Applications variables when files are detected in an import directory. The stored procedure that processes the lines of the import file should do end-of-file checking, as it will be called once per line, and will be passed the string EOF when the EOF character is hit at the bottom of the text file.

```
if exists (select * from sysobjects where id =
object_id('dbo.spLocal_DelimitedImport') and sysstat & 0xf = 4)
    drop procedure dbo.spLocal_DelimitedImport
```

GO

```
CREATE PROCEDURE dbo.spLocal_DelimitedImport
@ReturnStatus int OUTPUT,
@ReturnMessage varchar(255) OUTPUT,
@User_Id int,
@EC_Id int,
@RollNumber varchar(25),
@VarName varchar(25),
@Value varchar(25),
@Qualifier varchar(25)
```

As

```
Declare
@PU_Id int,
@PU_Desc varchar(50),
@Event_Id int,
@EventTimestamp datetime,
@NumSecs int,
@RawTimestamp datetime,
@Extended_Info varchar(3),
@Var_Id int,
@RawVar_Id int,
@RawPU_Id int,
@RawVar_IdString varchar(5),
@Var_IdString varchar(5),
@VUVar_Id int,
@VUPU_Id int,
@VUUser_Id int,
@VUCanceled int,
@VUResult Varchar(25),
```



## Models

```
@VUResult_On varchar(25),
@VUTransaction_Type int,
@VUPostUpdate int

Select @ReturnStatus = 0
Select @ReturnMessage = ''
Select @PU_Id = NULL
Select @Event_Id = Null
Select @EventTimestamp = NULL
Select @NumSecs = NULL
Select @RawTimestamp = NULL
Select @Extended_Info = NULL
Select @Var_Id = NULL
Select @RawVar_Id = NULL
Select @RawPU_Id = NULL
Select @RawVar_IdString = NULL
Select @Var_IdString = NULL
Select @VUVar_Id = Null
Select @VUPU_Id = Null
Select @VUUser_Id = Null
Select @VUCanceled = Null
Select @VUResult = Null
Select @VUResult_On = Null
Select @VUTransaction_Type = Null
Select @VUPostUpdate = Null

Select @RollNumber = LTrim(RTrim(@RollNumber))
Select @VarName = LTrim(Rtrim(@VarName))
Select @Value = LTrim(Rtrim(@Value))
Select @Qualifier = LTrim(Rtrim(@Qualifier))

If @RollNumber = 'EOF'
    BEGIN
        Select @ReturnMessage = 'spLocal_DelimitedImport-EOF'
        Goto ErrorOut
    END

If (@RollNumber = '') And (@VarName = '')
    BEGIN
        Select @ReturnMessage = 'spLocal_DelimitedImport-Blank Row'
        Goto NoProcessing
    END

If (@RollNumber = Null) And (@VarName = Null)
```

```

BEGIN
    Select @ReturnMessage = 'spLocal_DelimitedImport-All Values Null'
    Goto ErrorOut
END

If @RollNumber = 'Sample ID'
    BEGIN
        Select @ReturnMessage = 'spLocal_DelimitedImport,Sample Id Row for
EC_Id =' + Convert(varchar(4),@EC_Id)
        Select @ReturnStatus = 1
    END

Select @PU_Id = PU_Id from Event_Configuration
    Where EC_Id = @EC_Id
If @PU_Id IS NULL
    BEGIN
        Select @ReturnMessage = 'spLocal_DelimitedImport,Failure-PU_Id not
found for EC_Id =' + Convert(varchar(4),@EC_Id)
        Goto ErrorOut
    END

Select @PU_Desc = PU_Desc from Prod_Units
    where PU_Id = @PU_Id

CREATE TABLE #VariableUpdates (
    VUVar_Id int,
    VUPU_Id int,
    VUUser_Id int,
    VUCanceled int,
    VUResult Varchar(25),
    VUResult_On varchar(25),
    VUTransaction_Type int,
    VUPostUpdate int,
)

--Select Production Unit Where Raw Data goes
Select @RawPU_Id = Convert(int,Extended_Info) from Prod_Units
    where PU_Id = @PU_Id

Select @RawVar_Id = Var_Id from Variables
    where PU_Id = @RawPU_Id and Extended_Info = 'ST:' + @VarName

If @RawVar_Id IS NULL
    BEGIN

```

## Models

```
        Select @RawVar_IdString = 'Not Found'
        Select @ReturnMessage = 'Unit-' + @PU_Desc +
',spLocal_DelimitedImport,Failure-RawVar_Id not found for Extended_Info=' +
'ST:' + @VarName + ',PU_Id-' + Convert (varchar(5),@RawPU_Id)
        Goto ErrorOut
    END
Else
    Begin
        Select @RawVar_IdString = Convert (varchar(5),@RawVar_Id)
    End

Select @Var_Id = Var_Id from Variables
    where PU_Id = @PU_Id and Extended_Info = 'ST:' + @VarName
If @Var_Id IS NULL
    BEGIN
        Select @Var_IdString = 'Not Found'
        Select @ReturnMessage = 'Unit-' + @PU_Desc +
',spLocal_DelimitedImport,Failure-Var_Id not found for Extended_Info=' +
'ST:' + @VarName + ',PU_Id-' + Convert (varchar(5),@PU_Id)
        Goto ErrorOut
    END
Else
    Begin
        Select @Var_IdString = Convert (varchar(5),@Var_Id)
    End

If (@RollNumber = '') Or (@RollNumber Is Null) Or (@RollNumber =
'CFR12001018042')
    Begin
        --Trim Seconds from CurrentTime
        Select @RawTimestamp = GetDate()
        Select @NumSecs = DatePart(Second, @RawTimestamp)
        If (@NumSecs > 0)
            Begin
                Select @RawTimestamp = DateAdd(Second, - @NumSecs, @RawTimeStamp)
            End
    End
If @RawVar_Id Is Not Null
    Begin
        Select @VUVar_Id = @RawVar_Id
        Select @VUPU_Id = @RawPU_Id
        Select @VUUser_Id = 6
        Select @VUCanceled = 0
        Select @VUResult = @Value
        Select @VUResult_On = @RawTimeStamp
        Select @VUTransaction_Type = 1
```

```

Select @VUPostUpdate = 0

Insert into #VariableUpdates
(VUVar_Id,VUPU_Id,VUUser_Id,VUCanceled,VUResult,VUResult_On,VUTransaction_T
ype,VUPostUpdate)

Values(@VUVar_Id,@VUPU_Id,@VUUser_Id,@VUCanceled,@VUResult,@VUResult_On,@VU
Transaction_Type,@VUPostUpdate)

-- Insert Local_VariableUpdates
(VUVar_Id,VUPU_Id,VUUser_Id,VUCanceled,VUResult,VUResult_On,VUTransaction_T
ype,VUPostUpdate)
-
- Values(@VUVar_Id,@VUPU_Id,@VUUser_Id,@VUCanceled,@VUResult,@VUResult
t_On,@VUTransaction_Type,@VUPostUpdate)
End
End
Else
Begin
Select @Event_Id = Event_Id, @EventTimestamp = Timestamp from Events
Where PU_Id = @PU_Id and Event_Num = @RollNumber
If @Event_Id Is Null
BEGIN
Select @ReturnMessage = 'Unit-' + @PU_Desc +
',spLocal_DelimitedImport,Event Not Found for ' + @RollNumber + ',Raw
Processed RawVar_Id=' + @RawVar_IdString + ',Result-' + @Value
Goto RawOnly

END

If (@RawVar_Id Is Not Null) And (@EventTimeStamp Is Not Null)
Begin
Select @VUVar_Id = @RawVar_Id
Select @VUPU_Id = @RawPU_Id
Select @VUUser_Id = 6
Select @VUCanceled = 0
Select @VUResult = @Value
Select @VUResult_On = @EventTimeStamp
Select @VUTransaction_Type = 1
Select @VUPostUpdate = 0

Insert into #VariableUpdates
(VUVar_Id,VUPU_Id,VUUser_Id,VUCanceled,VUResult,VUResult_On,VUTransaction_T
ype,VUPostUpdate)
Values(@VUVar_Id,@VUPU_Id,@VUUser_Id,@VUCanceled,@VUResult,@VUResult_
On,@VUTransaction_Type,@VUPostUpdate)
-- Insert Local_VariableUpdates
(VUVar_Id,VUPU_Id,VUUser_Id,VUCanceled,VUResult,VUResult_On,VUTransaction_T
ype,VUPostUpdate)

```

## Models

```
-
-      Values(@VUVar_Id,@VUPU_Id,@VUUser_Id,@VUCanceled,@VUResult,@VUResult_On,@VUTransaction_Type,@VUPostUpdate)
      End

      If @Var_Id Is Not Null
      Begin
          Select @VUVar_Id = @Var_Id
          Select @VUPU_Id = @PU_Id
          Select @VUUser_Id = 6
          Select @VUCanceled = 0
          Select @VUResult = @Value
          Select @VUResult_On = @EventTimeStamp
          Select @VUTransaction_Type = 1
          Select @VUPostUpdate = 0

          Insert into #VariableUpdates
          (VUVar_Id,VUPU_Id,VUUser_Id,VUCanceled,VUResult,VUResult_On,VUTransaction_Type,VUPostUpdate)
          Values(@VUVar_Id,@VUPU_Id,@VUUser_Id,@VUCanceled,@VUResult,@VUResult_On,@VUTransaction_Type,@VUPostUpdate)

--      Insert Local_VariableUpdates
--      (VUVar_Id,VUPU_Id,VUUser_Id,VUCanceled,VUResult,VUResult_On,VUTransaction_Type,VUPostUpdate)
-
-      Values(@VUVar_Id,@VUPU_Id,@VUUser_Id,@VUCanceled,@VUResult,@VUResult_On,@VUTransaction_Type,@VUPostUpdate)
      End
      End

Select
2,VUVar_Id,VUPU_Id,VUUser_Id,VUCanceled,VUResult,VUResult_On,VUTransaction_Type,VUPostUpdate from #VariableUpdates
If @EventTimeStamp Is Not Null
    Select @ReturnMessage = 'Unit-' + @PU_Desc +
    ',spLocal_DelimitedImport,Successful Variable Update for ' + 'ST:' +
    @VarName + ',Var_Id=' + @Var_IdString + ',Time-' +
    Convert (varchar(20),@EventTimeStamp) + ',RawVar_Id=' + @RawVar_IdString +
    ',Result-' + @Value
--    Insert Local_ModelChecks
--    (VUVar_Id,Value1,Value2,Value3,Value4,ReturnMessage)
--    Values(@Var_Id,@RollNumber,@VarName,@Value,@Qualifier,@ReturnMessage)
If @EventTimeStamp Is Null
    Select @ReturnMessage = 'Unit-' + @PU_Desc +
    ',spLocal_DelimitedImport,Successful Variable Update for ' + 'ST:' +
    @VarName + ',RawVar_Id=' + @RawVar_IdString + ',Result-' + @Value + ',Time-' +
    Convert (varchar(20),@RawTimeStamp)
--    Insert Local_ModelChecks
--    (VUVar_Id,Value1,Value2,Value3,Value4,ReturnMessage)
```

```

-
- Values (@RawVar_Id,@RollNumber,@VarName,@Value,@Qualifier,@ReturnMessage)
-
Select @ReturnStatus = 1
Return

RawOnly:
Select
2,VUVar_Id,VUPU_Id,VUUser_Id,VUCanceled,VUResult,VUResult_On,VUTransaction_
Type,VUPostUpdate from #VariableUpdates
--Insert Local_ModelChecks
(VUVar_Id,Value1,Value2,Value3,Value4,ReturnMessage)
-
- Values (@RawVar_Id,@RollNumber,@VarName,@Value,@Qualifier,@ReturnMessage)
-
Select @ReturnStatus = 1
Return

NoProcessing:
Select @ReturnStatus = 1
Return

ErrorOut:
Select @ReturnStatus = 0
--Insert Local_ModelChecks
(VUVar_Id,Value1,Value2,Value3,Value4,ReturnMessage)
-- Values (0,@RollNumber,@VarName,@Value,@Qualifier,@ReturnMessage)
Return

GO

GRANT EXECUTE ON dbo.spLocal_DelimitedImport TO ComXClient
GO

```

## Model 84

Model 84 is a test data import interface model. It imports Sheeter data from a specific file format for a specific site.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 84 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |

## Models

|                           |   |
|---------------------------|---|
| Model Processing Group    | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information. |
| TINT:Interval (Minutes)   | Timing Interval the model checks for New Import Files.                                      |
| Import File Location Spec | Drive and directory for Import Files.   |
| Accepted File Path        | Drive and directory for Successfully Processed Files.                                       |
| Rejected File Path        | Drive and directory for Rejected Files.   |
| Purge Days                | # Days of Accepted/Rejected files to keep   |

### Model 85

Model 85 is a test data import interface model. It imports Beater data from a specific file format for a specific site.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

#### Model 85 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Grade Change (TRUE)               | Create Grade Changes from the file. The default is True.   |
| Purge Days                        | # Days of Accepted/Rejected files to keep  |

### Model 86

Model 86 is a test data import interface model. It imports Roll data from a specific file format for a specific site.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

#### Model 86 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |

|                     |  |
|---------------------|--|
| Grade Change (TRUE) | Create Grade Changes from the file. The default is True. |
| Purge Days          | # Days of Accepted/Rejected files to keep                |

## Model 87

Model 87 imports test data from an external system after every interval is elapsed. The data is retrieved from an external database using an ODBC data source. The data is imported and removed from Plant Applications using stored procedures.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

**IMPORTANT:** *The ODB connection must NOT have the parameter "Enable Lazy Close Support" selected to avoid locking the Plant Applications tables.*

---

### Model 87 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:(Interval) Minutes           | A value change of this tag is used to trigger a new production event to be created.  |
| ODBC Connect String               | DSN=xx;UID=yyy; PWD=zzz;   |
| SpName (NewTests)                 |  |
| SpName (RemoveTest)               |  |
| Misc                              |  |

## Model 91

Model 91 is a downtime data import interface model. It imports Downtime or Delay data from a specific file format. This model is set up for a specific site.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 91 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Location                          | Delay Location.  |
| Default Fault                     | Delay Fault.   |



|            |  |
|------------|--|
| Purge Days | # Days of Accepted/Rejected files to keep. |
|------------|--|

## Configure Model 5013 for Batch Analysis

Model 5013 is used to extract the data from your batch journal table (sometimes known as the Batchhis table) and write them to the Event\_Transactions table. This model needs to be configured on the production unit that you are using as your batching unit.

If you are not using Proficy Batch Execution or RSBatch, you must find another method for populating the Event\_Transactions table, such as creating your own custom stored procedure and configuring a Model 602 to call your custom stored procedure. For more information on configuring a Model 602, see the topic, [Generic Model 602-Interval Triggered](#).

### Using RSBatch

If you are using RSBatch, Model 5013 will fire on the interval specified when configuring the model and look to see if there are any unprocessed batches in the Local\_BatchComplete table. If it identifies unprocessed batches, it will then select all the records from the Batchhis table, perform the required data mapping, insert the records into the Event\_Transactions table, and mark the batch as processed in the Local\_BatchComplete table.

- The RSBatch interface executes at the 'End of Batch'; it does not process batch records real-time.
- Model 5013 will process the maximum number of batches per cycle specified in the (**rsBatch**) **Max Batches to Process per run** option (the default is 1), ordered by lcltime.

### Using Proficy Batch Execution

For Proficy Batch Execution, Model 5013 will fire at the specified interval and attempt to process any unprocessed records from the Batch Analysis table, based on the specified parameters.

To configure Model 5013:

**Plant Applications Administrator > Plant Model > Department > Production Line > Production Unit**

1. Ensure you have configured Model 118. For more information, see the topic [Configure Model 118 for Batch Analysis](#).
2. Right-click the production unit you are using as your batching unit and click **Configure Events on <production unit>**. The **Event Configuration** wizard appears.
3. On the **Configured Models** tab, select **Import** from the Model Type list and click **Add New Model**. The **General** tabbed page is added and becomes active.
4. Click **More Models**. The **Search** dialog box appears.
5. Under **Model Number**, select model **5013** and click **OK**.
6. On the **General** tab, edit the following options:
  - a. **optional:** In the **Exclusions** box, type the values this model is to ignore and not use to trigger the model. Use a comma to separate values.
  - b. **optional:** In the **Extended Information** box, type additional information that may be useful. This is not used on any reports or displays.
  - c. **optional:** In the **Maximum Run Time (Seconds)** box, type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop).

- d. **optional:** Type the group number in the Model Processing Group box. The Model Processing Group number can be up to six digits.
- e. The **Model Processing Group** field is not used for this model.
- f. In the **TINT: Interval (Minutes)** box, type the timing interval, in minutes, between calls.
- g. In the **spName** box, select the correct stored procedure (**spS88\_iBatchReader** or **spS88\_RSBatch6Reader**).
- h. In the **Delay(Seconds)** box, type the number of seconds to add to the timestamp. Typically, this is used to adjust for data that falls outside the sampling window because Plant Applications cannot recognize millisecond resolution.

---

**IMPORTANT:** Do not use this field unless you have advanced knowledge of SQL and Plant Applications.

---

- i. The three **Reserved** fields are reserved for future use.
- j. *Proficiency Batch Execution only* — In the **(iBatch) Field for Batch Instance** box, type a unique ID to use for creating the batch name. If used, the batch name would be **batch name | batch instance**. This can be left blank.
- k. *Proficiency Batch Execution only* — In the **(iBatch) Field for Batch Name** box, type the batch ID or a unique ID, which is used for building the batch name. The batch name will be **batch name | batch instance** (if the batch Instance is specified).
- l. *Proficiency Batch Execution only* — Select the **(iBatch) Ignore Ready and Idle Status** option to ignore records that have a StateValue of 'Ready' or 'Idle'. These records will not be inserted into the Event\_Transactions table. Selecting this option will prevent Ready and Idle records that occur after the batch complete state from giving a false end-of-batch time.
- m. *Proficiency Batch Execution only* — In the **(iBatch) Max records to process per run** box, type the maximum number of records to process per stored procedure call. The default is 5000 records per stored procedure call.
- n. *Proficiency Batch Execution only* — In the **(iBatch) Unique Field Name** box, type the name for the column that contains the record order for the batch. The default is "Sequence".
- o. *RSBatch only* — In the **(rsBatch) Filter** box, specify a filter that will remove records that meet the criteria prior to insertion into the Event\_Transactions table. It must reference the field names from the Event\_Transactions table. This is an example of a filter:

```
(ParameterAttributeValue = '-9999' OR RecipeString LIKE '%$NULL%'
OR ((ParameterAttributeValue IS NULL OR ParameterAttributeValue =
'') AND EventName IN ('Recipe Header', 'Report', 'Scale Factor',
'Param Download Verified', 'Recipe Value')))
```

- p. *RSBatch only* — In the **(rsBatch) Max Batches to Process per run** box, type the number of batches to process. Previously, processing was done by number of records.
- q. *RSBatch only* — In the **(rsBatch) Table Name for Completed Batches** box, type the name of the Local\_BatchComplete table that is created on the batch history database. If you are running multiple interfaces from the same Plant Applications server, you must have multiple, uniquely named Local\_BatchComplete tables.
- r. In the **Batch Database Name** box, type the name of the batch history database. If the database is on another server (a linked server), use the format **servername.databasename**.

## Models

- s. In the **Batch Database Owner** box, type the name of the SQL database owner of the batch history database. The default owner is 'dbo'.
- t. In the **Batch Table Name** box, do one of the following:
  - o For RSBATCH, type the name of the table that contains the batch journal records.
  - o For Proficy Batch Execution, you must use "BatchAnalysis".
- u. In the **Decimal Separator on Batches** box, type either a comma or period. If the source batch journal data is stored with a comma, it will be converted to a decimal before insertion into the Event\_Transactions table.
- v. In the **Group Separator on Batches** box, type the group separator that will be replaced with a comma prior to insertion into the Event\_Transactions table.
- w. In the **Product Code Parameter Name** box, type the name in the Descript field that contains the product for the batch
- x. Select the **Recipe Value in ParameterReport** option to convert the event types of 'Recipe Value' to ParameterReport Eventtypes in Event Transactions. They will be treated as variable data instead of specifications.
- y. Select the **Use Batch Instance** option to use the Batch Instance field when building the Batchname with the format of BatchName|BatchInstance.

12. Click  **Save**.

## Model 1053-Consumption Model

Consumption Models are used to update the amount of input events have been consumed into the Raw Material Inputs on a Producing Unit. Consumption Models are triggered by additions to the Event\_Components table for any valid source units that are defined for the Raw Material Inputs defined on the Producing Unit. A Consumption Model is specific to the Raw Material Inputs. Therefore, a given unit may have several Raw Material Inputs and multiple Consumption Models active. Typical operations programmed into the Stored Procedure defined for a Consumption Model are:

- Update the amount of the input events that have been consumed to produce the Product on the Producing Unit.
- Update Event Statuses to indicate Partial or Total Consumption of an Event.

---

*The functionality in this model is available only if you have purchased the Production Management Module.*

---

### Consumption Model 1053 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Local SP Name                     | Stored Procedure called when the model is triggered.   |

## Input Genealogy Models

Input Genealogy Models are used to create and attach events together as event components as product is moving through Raw Material Inputs on a Producing Unit. They are triggered when changes are made to Production Events and to the position of Production Events on Producing Units. These changes can occur manually in the Genealogy View or in AutoLog. These changes can also occur from Historian Tag value changes or Plant Applications Variable updates. An Input Genealogy Model is specific to the Raw Material Inputs. Therefore, a given unit may have several Raw Material Inputs and multiple Movement Models active. Typical operations programmed into the Stored Procedure defined for a Movement Model are:

- Attach event components when links are made between events.
- Un-attach event components when links are broken between events.
- Create child events when parent events are created.

The following tables cause Input Genealogy Models to be triggered:

Table = PrdExec\_Input\_Event

- An Event is Staged
- An Event moves to the Running position

Table = PrdExec\_Input\_Event\_History

- An Event is Staged
- An Event Moves to the Running Position
- Event in the Running Position is Completed
- A Staged Event is Unloaded
- A Running Event is Unloaded

Table = Events

- New Event is created
- Changes are made to an Event. For example when the End Time of an Event is changed or if the Event Number is changed.
- Status of an Event is changed.

### Model 1052-Input Genealogy Model

---

*The functionality in this model is available only if you have purchased the Production Management Module.*

---

#### Input Genealogy Model 1052 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |

## Models

|                        |   |
|------------------------|---|
| Model Processing Group | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information. |
| Local SP Name          | Stored Procedure called when the model is triggered.  |

### Movement Model 1052 Stored Procedure Parameters

The following parameters are required at the beginning of the Stored Procedure that is called by Model 1051. The parameter variables can be named whatever is desired but the order must be maintained.

| Variable Name                       | Parameter Description   |
|-------------------------------------|---|
| @ReturnStatus int OUTPUT,           | Flag to indicate Success or Failure (1-Success,0-Failure)   |
| @ReturnMessage varchar(255) OUTPUT, | Error Message to Write to Log File  |
| @EC_Id int,                         | Input to the Stored Procedure indicating the EC_Id of the model from the Event_Configuration table. |
| @TableName varchar(255),            | Input to the Stored Procedure indicating the Table Name that triggered the Model.                   |
| @Id                                 | Input to the Stored Procedure indicating the Id from the record that triggered the Model.           |

## Model 5008 - Input Genealogy Model

### Attach When Running (Based on Time)

Input Genealogy Model 5008 links an event that moves into the Running position to the Event that is on the output side. It looks at the last event on the output side, based on time, and links it to the event that was moved into the Running position.

### Input Genealogy Model 5008 Properties

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| spLocal - Stored Procedure Name   | spLocal_Sample5008: This is the stored procedure called when the model is triggered.   |

### Input Genealogy Model 5008 Stored Procedure Parameters

This Stored Procedure is pre-configured.

**IMPORTANT:** You can edit this stored procedure. However, if you do edit it, you must rename it; otherwise, on the next Plant Applications upgrade, the stored procedure will get overwritten and you will lose any changes that you made.

## Model 5009 - Input Genealogy Model

### Match to Event\_Num on Output Side

When an Event moves to the Running position, Model 5009 looks up the Event Number and finds the Event Number on the output side and links the Event in the Running position to the Event on the output side, based on the Event Number.

### Input Genealogy Model 5009 Properties

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| spLocal - Stored Procedure Name   | spLocal_Sample5009: This is the stored procedure called when the model is triggered.   |

### Input Genealogy Model 5009 Stored Procedure Parameters

This Stored Procedure is pre-configured.

**IMPORTANT:** You can edit this stored procedure. However, if you do edit it, you **must rename it**; otherwise, on the next Plant Applications upgrade, the stored procedure will get overwritten and you will lose any changes that you made.

## Model 5010 - Input Genealogy Model

### Create Output Event Based on Input

When an Event moves into the Running position, an Event with the same number as the Event in the Running position is created on the output unit and the two Events are linked together.

### Input Genealogy Model 5010 Properties

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| spLocal - Stored Procedure Name   | spLocal_Sample5010: This is the stored procedure called when the model is triggered.   |

### Input Genealogy Model 5010 Stored Procedure Parameters

This Stored Procedure is pre-configured.

**IMPORTANT:** You can edit this stored procedure. However, if you do edit it, you **must rename it**; otherwise, on the next Plant Applications upgrade, the stored procedure will get overwritten and you will lose any changes that you made.

## Input Movement Models

Movement Models are used to move items into Raw Material Input positions and out of Raw Material Input positions on a Producing Unit. This input item can indicate a new event (batch of raw material) is being used as an input to the product being created on an output or producing unit. It can indicate that an item has been consumed and is now completed running on the Producing Unit and it produces the required outputs for that unit.

## Models

Movement Models are triggered by Historian Tag value changes or Plant Applications Variable updates. When a Historian Tag value changes or a Plant Applications Variable is updated the Movement Model calls the defined Stored Procedure. A Movement Model is specific to the Raw Material Inputs. Therefore, a given unit may have several Raw Material Inputs and multiple Movement Models active. Typical operations programmed into the Stored Procedure defined for a Movement Model are:

- Move an event from Inventory into an Input Position such as Staged or Running.
- Move an event from Staged to Running.
- Move an event from Running to Complete.
- Create Output events.
- Make Input and Output Event Status changes.

Other Models available that could be substituted in place of the standard Movement Model are Models 600 and 603. Model 600 watches a single Historian Tag value for changes and calls the defined Stored Procedure calculation. Model 603 watches multiple Historian Tags for any value changes and then calls the defined Stored Procedure.

### Model 1051- Input Movement Model

---

**NOTE:** The functionality in this model is available only if you have purchased the Production Management Module.

---

#### Movement Model 1051 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                   | Description  |
|----------------------------|--|
| Maximum Run Time (Seconds) | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop).   |
| Model Processing Group     | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TriggerTag(s)              | The tag or tags used to trigger the model. A value change on any of the defined tags will trigger the model and the Stored Procedure is called. In the Plant Applications Administrator, a Value from this tag is passed into the Stored Procedure based on the chosen Sampling Type.  |
| Trigger Var Id(s)          | The variables or variables used to trigger the model. A value change on any of the defined variables will trigger the model and the Stored Procedure is called and the Variable Value is passed into the Stored Procedure.   |
| Local SP Name              | Stored Procedure called when the model is triggered.   |
| Value Change Only          | If the Value Change Only is <b>false</b> (this is the default) then the model is triggered by new historian tag values for any of the tags defined. When a new historian tag value is available for any tag, the model calls the defined stored procedure. <b>The value does not have to change for the model to be triggered.</b><br><br>If the Value Change Only is <b>true</b> then the model is triggered only by a change to consecutive historian tag archive values for any of the tags defined. When a value change is detected, the model calls the defined stored procedure. <b>The value must change for the model to be triggered.</b> |

### Movement Model 1051 Stored Procedure Parameters

The following parameters are required at the beginning of the Stored Procedure that is called by Model 1051. The parameter variables can be named whatever is desired but the order must be maintained.

| Variable Name                       | Parameter Description  |
|-------------------------------------|--|
| @ReturnStatus int OUTPUT,           | Flag to indicate Success or Failure (1-Success,0-Failure)  |
| @ReturnMessage varchar(255) OUTPUT, | Error Message to Write to Log File   |
| @JumpToTime Datetime OUTPUT,        | This is used to allow you to control how far back the EventManager goes in time to look for tag changes after a reload or restart of the Event Manager. By default the Event Manager goes back 3 days and looks for tags changes. In your code you could find the latest event on your unit and set the JumpToTime to be a second after this and then have it start looking from there for new events. |
| @EC_ID int,                         | Input to the Stored Procedure indicating the EC_Id of the model from the Event_Configuration table.  |
| @ReservedInput1 varchar(30),        | Reserved   |
| @ReservedInput2 varchar(30),        | Reserved   |
| @ReservedInput3 varchar(30),        | Reserved   |
| @ReservedInput4 varchar(30),        | Reserved   |
| @TriggerTag varchar(50),            | The Alias of the Tag which Triggered the SP  |
| @TriggerTag_OldValue varchar(25),   | The Previous Value of the Triggering Tag   |
| @TriggerTag_OldTime Datetime,       | The Previous Time of the Triggering Tag  |
| @TriggerTag_NewValue varchar(25),   | The New Value of the Triggering Tag  |
| @TriggerTag_NewTime Datetime,       | The New Time of the Triggering Tag   |
| @Hist1Alias varchar(30),            | The Alias Letter For Tag or Variable 1   |
| @Hist1Value varchar(30),            | The Value For Tag or Variable 1  |
| @Hist2Alias varchar(30),            | The Alias Name For Tag or Variable 2   |
| @Hist2Value varchar(30)             | The Value For Tag or Variable 2  |
| ...continued as needed              |  |

### Model 5002- Move Oldest Event to Staged (Historian Tag)

When the value in the trigger tag changes, Model 5002 moves the oldest Event to the Staged position. The model first looks at the Running position and if there is something in the Running position moves it to Completed. Next, it looks to see if any Event is in the Stage position, and if there is, moves it to the Running position. Finally, it loads the triggered event into Running, if there is no event in Running, or moves it into Staged, if there is an Event in Running.

#### Input Movement Model 5002 Properties

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |



|               |   |
|---------------|---|
| PT:TriggerTag | The tag used to trigger the model. A value change on any of the defined tags will trigger the model and the Stored Procedure is called. |
| Local SP Name | spLocal_Sample5002: The pre-configured stored procedure called when Model is triggered.   |

### Input Movement Model 5002 Stored Procedure Parameters

This stored procedure is pre-configured.

**IMPORTANT:** You can edit this stored procedure. However, if you do edit it, you **must rename it**; otherwise, on the next Plant Applications upgrade, the stored procedure will get overwritten and you will lose any changes that you made.

### Model 5003 - Move Oldest Event to Staged (Variable)

When the value in the variable changes, Model 5003 moves the oldest Event to the Staged position. The model first looks at the Running position and if there is something in the Running position moves it to Completed. Next, it looks to see if any Event is in the Stage position, and if there is, moves it to the Running position. Finally, it loads the triggered event into Running, if there is no event in Running, or moves it into Staged, if there is an Event in Running.

#### Input Movement Model 5002 Properties

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Variable                          | The variable used to trigger the model. A value change on any of the defined variable will trigger the model and the Stored Procedure is called and the Variable Value is passed into the Stored Procedure.  |
| Local SP Name                     | spLocal_Sample5003: The pre-configured stored procedure called when Model is triggered.  |

### Input Movement Model 5003 Stored Procedure Parameters

This Stored Procedure is pre-configured.

**IMPORTANT:** You can edit this stored procedure. However, if you do edit it, you **must rename it**; otherwise, on the next Plant Applications upgrade, the stored procedure will get overwritten and you will lose any changes that you made.

### Model 5004 - By Event Number (Historian Tag)

When the value in the trigger tag changes, Model 5004 moves the Event to the Staged position. The model first looks at the Running position and if there is something in the Running position moves it to Completed. Next, it looks to see if any Event is in the Stage position, and if there is, moves it to the Running position. Finally, it loads the triggered event into Running, if there is no event in Running, or moves it into Staged, if there is an Event in Running.

#### Input Movement Model 5004 Properties

| Property | Description |
|----------|-------------|
|----------|-------------|

|                                   |  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| PT:TriggerTag                     | The tag containing the Event_Num and used to trigger the model. A value change on any of the defined tags will trigger the model and the Stored Procedure is called.   |
| Local SP Name                     | spLocal_Sample5004: The pre-configured stored procedure called when Model is triggered.  |

### Input Movement Model 5004 Stored Procedure Parameters

This Stored Procedure is pre-configured.

**IMPORTANT:** You can edit this stored procedure. However, if you do edit it, you **must rename it**; otherwise, on the next Plant Applications upgrade, the stored procedure will get overwritten and you will lose any changes that you made.

## Model 5005 - By Event Number (Variable)

### By Event\_num - Variable

When the value in the variable changes, Model 5005 moves the Event to the Staged position. The model first looks at the Running position and if there is something in the Running position moves it to Completed. Next, it looks to see if any Event is in the Stage position, and if there is, moves it to the Running position. Finally, it loads the triggered event into Running, if there is no event in Running, or moves it into Staged, if there is an Event in Running.

### Input Movement Model 5005 Properties

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Variable                          | The ID of the variable containing the Event_Num and used to trigger the model. A value change will trigger the model and the Stored Procedure is called.   |
| Local SP Name                     | spLocal_Sample5005: The pre-configured stored procedure called when Model is triggered.  |

### Input Movement Model 5005 Stored Procedure Parameters

This Stored Procedure is pre-configured.

**IMPORTANT:** You can edit this stored procedure. However, if you do edit it, you **must rename it**; otherwise, on the next Plant Applications upgrade, the stored procedure will get overwritten and you will lose any changes that you made.

## Model 5006 - By Event Number and Unit

## Models

When the value in the trigger tag changes, Model 5006 moves the Event to the Staged position. The model first looks at the Running position and if there is something in the Running position moves it to Completed. Next, it looks to see if any Event is in the Stage position, and if there is, moves it to the Running position. Finally, it loads the triggered event into Running, if there is no event in Running, or moves it into Staged, if there is an Event in Running.

### Input Movement Model 5004 Properties

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| PT:TriggerTag                     | You can select two historian tags as trigger tags. The first tag contains the event number and the second tag contains the production unit ID (PU_ID). A value change will trigger the model and the stored procedure is called.   |
| Local SP Name                     | The custom (spLocal) stored procedure called when the model is triggered. The default is spLocal_Sample5006:   |

### Input Movement Model 5006 Stored Procedure Parameters

The default stored procedure is spLocal\_Sample5006. You can select any spLocal stored procedure.

**IMPORTANT:** You can edit the spLocal\_Sample5006 stored procedure. However, if you do edit it, you **must rename it**; otherwise, on the next Plant Applications upgrade, the stored procedure will get overwritten and you will lose any changes that you made.

## Model 5007 - By Event Number and Unit (Variables)

When the value in the variable changes, Model 5007 moves the Event to the Staged position. The model first looks at the Running position and if there is something in the Running position moves it to Completed. Next, it looks to see if any Event is in the Stage position, and if there is, moves it to the Running position. Finally, it loads the triggered event into Running, if there is no event in Running, or moves it into Staged, if there is an Event in Running.

### Input Movement Model 5005 Properties

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Variable                          | The ID of the variable containing the event number and the ID of the variable containing the production unit (PU_ID) used to trigger the model. A value change will trigger the model and the Stored Procedure is called.  |
| Local SP Name                     | spLocal_Sample5007: The pre-configured stored procedure called when Model is triggered.  |

### Input Movement Model 5005 Stored Procedure Parameters

This Stored Procedure is pre-configured.

**IMPORTANT:** You can edit this stored procedure. However, if you do edit it, you **must rename it**; otherwise, on the next Plant Applications upgrade, the stored procedure will get overwritten and you will lose any changes that you made.

## Process Order Models

### Model 1055-Schedule Change Model

A schedule change event occurs when additions, updates, or deletions are made to the production plan. These could be manual changes or changes made via an interface from an external scheduling system fed into Plant Applications.

---

*The functionality in this model is available only if you have purchased the Production Management Module.*

---

#### Schedule Change Model 1055 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Local SP Name                     | Stored Procedure called when the model is triggered.   |

#### Movement Model 1055 Stored Procedure Parameters

The following parameters are required at the beginning of the Stored Procedure that is called by Model 1055. The parameter variables can be named whatever is desired but the order must be maintained.

| Variable Name                       | Parameter Description   |
|-------------------------------------|---|
| @ReturnStatus int OUTPUT,           | Flag to indicate Success or Failure (1-Success,0-Failure)   |
| @ReturnMessage varchar(255) OUTPUT, | Error Message to Write to Log File  |
| @EC_ID int,                         | Input to the Stored Procedure indicating the EC_Id of the model from the Event_Configuration table. |
| @TableName varchar(255),            | Input to the Stored Procedure indicating the Table Name that triggered the Model.                   |
| @Id                                 | Input to the Stored Procedure indicating the Id from the record that triggered the Model.           |

### Creating Process Orders From Tags

Selecting **Process Order** from the **Model Type** list in the **Simple Event Configuration** wizard enables you to configure Model 804, which creates process orders from tags.







When this model is triggered, the model checks for duplicate process orders. If a duplicate process order is found, a suffix, in the form of **YYMMDDhhmmss** is added to the process order number. When the process order number is created, the process order is associated with the product code, if the code is read within the sampling window. New process orders are created with the status of **Pending**.

---

**NOTE:** The creation of a process order creates the production plan record **only**.

---

To create process orders from tags (Model 804):

1. In the  Plant Model, right-click the production unit where you want to detect events and click **Configure Events on <production unit>**. The **Event Detection** wizard appears.
2. On the **Configured Models** tab, select **Process Order** from the **Model Type** list and click **Add New Model**. The **General** tabbed page appears.
3. Do the following:
  - a. **optional:** In the **Maximum Run Time (Seconds)** box, type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop).
  - b. **optional:** Type the group number in the Model Processing Group box. The Model Processing Group number can be up to six digits.
  - c. **optional:** In the **Extended Information** box, type additional information that may be useful. This is not used on any reports or displays.
  - d. **optional:** In the **Exclusions** box, type the values this model is to ignore and not use to trigger the model. Use a comma to separate values.
  - e. In the **Process Order Tag** box, click the **Browse** button  to select the tag to generate the process order number.
  - f. In the **Product Code Tag** box, click the **Browse** button  to select the tag to generate the product code. If the product code does not exist, then the product is set to **None**.
  - g. In the **Product Code Window** box, type the number of seconds to define the sampling window. The maximum value is 86,400 seconds. If the product code tag is not populated at the same time as the process order tag, this number defines how far back or forward in time the program will look for a product tag change. The start time of the sampling window is determined by the time of the trigger tag.
  - h. In the **Quantity Tag** box, click the **Browse** button  to select the tag to specify the quantity.
  - i. In the **Execution Path** box, select a configured execution path. If no execution paths have been configured, the list will be empty. For information on configuring execution paths, please see Defining Execution Paths.
  - j. **optional:** In the Comments box, type any comments.
4. Click  to activate the model.
5. Click **Save** .

## Product Change Models

### Model 100

Model 100 creates product changes when the Grade Change Tag has a value change. The Add Minutes parameter value is added to the Timestamp of the value change to create the Timestamp for the Product Change. The value of the Tag is used to lookup the Product Code using the Product Cross Reference value.

**NOTE:** The functionality in this model is available only if you have purchased the Product Management Module.

### Model 100 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Grade Change Tag                  | A value change of this tag is used to trigger a product change to be created.  |
| Add Minutes (Optional)            | This value added to the Timestamp of the value change time.  |


### Detecting Product Change Events

Model 803 can be configured to detect product change events. When the PCodeTag tag changes, a new product change event is created with the product code coming from the value of the tag.

- If the product does not exist in the database, it will be created and mapped to the unit the event is on.
- If the product exists in the database, but it is not mapped to the unit, then it will be mapped to the unit.
- If the product exists and it is mapped to the unit, then a simple product change event will occur.
- If the unit is a schedule point and the Path Flag is true and the product does not exist on the path, the product will be created.

Product Change Model 803 rounds a numeric string to an integer and also trims leading zeros from strings containing all numeric characters. It uses whatever string it gets from an historian, numeric characters or otherwise.

To configure Model 803:

1. In the  Plant Model, right-click the production unit where you want to detect events and select **Configure Events on <production unit>**. The **Event Detection** wizard appears.
2. On the Configured Models tab, select Product Change from the Model Type list and click Add New Model.
3. Click the **General** tab and do the following:
  - a. **optional:** In the **Maximum Run Time (Seconds)** box, type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop).
  - b. **optional:** Type the group number in the Model Processing Group box. The Model Processing Group number can be up to six digits.
  - c. **optional:** In the **Extended Information** box, type additional information that may be useful. This is not used on any reports or displays.

- d. **optional:** In the **Exclusions** box, type the values this model is to ignore and not use to trigger the model.
  - e. Select **Automatically Create Missing Products** to create the product if the product does not exist.
  - f. **optional:** Select **Automatically Add Products To Paths** to add the product to the production execution path, if the unit is a schedule point. By default, the option is not selected and the product will not be assigned to any path.
  - g. **optional:** Select a default product family from the **Default Product Family** list. Products that were automatically created will be assigned to the selected product family. For more information, see Product Families.
4. Click the **Identify Input(s)** tab to identify and select the historian tags used for input.
    - a. If the tag is to be set as a trigger, select the **Trigger** check box. A minimum of one trigger tag is required.
    - b. Click the **Browse** button beside the **Tag** box. The **Tag Search** dialog box appears. Click **Search** (you can enter search criteria prior to clicking the button if necessary). Select the tag and click **OK**.
    - c. From the **Attribute** list, select either Value or Timestamp, depending on whether you want the value or time of the tag to be passed.
    - d. From the **Sampling Type** list, select the type of sampling that is applied to the tag. For more information on sampling types, see Sampling Types.
    - e. In the **Time Offset** box, enter the number of seconds backward from the trigger time that the next value will be retrieved.

For example: If you have trigger tag A in the list and you also have another tag B that is used as an input in the VB Script, you can specify what the time offset will be for tag B. This would be the amount of time backward from the trigger time that the EventMgr would attempt to get the 'Last Good Value' for tag B. So if tag A changed at 9/28/07 7:00, and you specified a 10 second offset for tag B, then the EventMgr would attempt to retrieve the Last Good Value for tag B starting at 9/28/07 6:59:50.

- f. In the Precision box, enter the number of digits to the right of the decimal point to use. This applies to numeric tag values only.




---

*NOTE: If the tag value contains only numeric values, it is interpreted as a number. Any digits after the decimal point are rounded to the specified precision, and any leading zeros are trimmed. If this is not desirable, the use the Product Code script to manipulate the parsing of numeric Product Codes. If the tag value contains alpha characters or otherwise cannot be parsed as a number, it is interpreted as a string.*


---

5. Click the **Scripts** tab to edit sample scripts or write script logic.

The following functionality is available on each tab.

- Click . The **Script Builder** dialog box appears where you can edit the existing script or to write script logic.
- In the **Script Builder** dialog box:
- Click . The **Select Alias** dialog box appears. Select the Alias and click **OK**. The Alias letter appears in the **Script** box.
- Click  **Check Syntax** to ensure you have entered the VB Script correctly.
- Click the **Define Product Code Script** tab to edit the VB Script. "ProductCode" is the required keyword. The script will be used to generate the product change event when

the model is triggered. It will support the expression: "ProductCode = Null" which will signify that there is no event to process and will prevent the model from firing.

6. Click  to activate the model.

## Production Event Models

### Model 1

Model 1 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the properties defined for this model. The format of the Tappi or Event Number is [?M?###]. The first question mark (?) refers to the first character of the Tappi or Event Number. The M refers to the Tappi Month characters defined. The second question mark (?) refers to the third character of the Tappi or Event Number. The ### refers to the event count of the month.

---

**NOTE:** *The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 1 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                   | Description  |
|----------------------------|--|
| Maximum Run Time (Seconds) | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group     | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                 | A value change of this tag is used to trigger a new production event to be created.  |
| 1st Char of Tappi          | This value is used as the 1st character of the Tappi or Event Number.  |
| 3rd Char of Tappi          | This value is used as the 3rd character of the Tappi or Event Number.  |
| Prod Day (HH:MM)           | The time that the counter is restarted for the number of events made in that day.  |
| Month Chars                | The Tappi Month Characters used to generate the M character.   |
| Use Prod Day (TRUE)        | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

### Model 2

Model 2 creates production events when the Turnup Tag value equals the value defined. The Tappi or Event Number is created using the properties defined for this model. The format of the Tappi or Event Number is [?MDD##]. The first question mark (?) refers to the first character of the Tappi or Event Number. The M refers to the Tappi Month characters defined. The second question mark (?) refers to the third character of the Tappi or Event Number. The ## refers to the event count of the day.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 2 Properties

The following Model properties are set up using the Plant Applications Administrator:



| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag triggers the creation of a new production event.  |
| The Value                         | The Value the Turnup Tag must equal to trigger the creation of a new Tappi or production event.  |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number.   |
| Month Chars                       | The Tappi Month Characters used to generate the M character.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |
| Common PU_Id (Optional)           | If a Common PU_Id is supplied then the Tappi or production event is also created on this common production unit.   |

### Model 3

Model 3 creates production events when the Batch Tag value transitions from a specific value to another specific value. The Event Number is created using the properties defined for this model. The format of the Event Number is [?YYMMDD###]. The 1st question mark (?) refers to the 1st character of the Event Number. The YY refers to a numeric 2 digit year. The MM refers to a numeric 2 digit month. The DD refers to a numeric 2 digit day of the month. The ## refers to the batch or event count of the day.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 3 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Batch Tag                         | The tag used to trigger the creation of a new production event.  |
| From Value                        | The Value the Batch Tag must transition FROM to trigger the creation of a new production event.  |
| To Value                          | The Value the Batch Tag must transition TO in order to trigger the creation of a new production event.   |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will  |

|  |   |
|--|---|
|  | impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day. |
|--|---|

## Model 4

Model 4 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the properties defined for this model. The format of the Tappi or Event Number is [?MDD##]. The first question mark (?) refers to the first character of the Tappi or Event Number. The M refers to the Tappi Month characters defined. The DD refers to a numeric 2 digit day of the month. The ## refers to the event count of the day.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 4 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Month Chars                       | The Tappi Month Characters used to generate the M character.   |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

## Model 5

Model 5 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the properties defined for this model. The format of the Tappi or Event Number is [?YMDD##]. The first question mark (?) refers to the first characters of the Tappi or Event Number. The M refers to the Tappi Month characters defined. The DD refers to a numeric 2 digit day of the month. The ## refers to the event count of the day.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 5 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |

|                                |   |
|--------------------------------|---|
| Model Processing Group         | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.   |
| Turnup Tag                     | A value change of this tag is used to trigger a new production event to be created.   |
| Prod Day (HH:MM)               | The time that the counter is restarted for the number of events made in that day.   |
| Tappi Begin                    | This value is used as the first characters of the Tappi or Event Number.  |
| Month Chars                    | The Tappi Month Characters used to generate the M character.  |
| Use Prod Day (TRUE)            | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day. |
| Check Tag (Optional)           | The Check Tag, Check Value, and Check Sampling Type are used to ignore a production event trigger if the Value of this tag using the Check Sampling Type is less than the defined Check Value.  |
| Check Value (Optional)         | The Check Tag, Check Value, and Check Sampling Type are used to ignore a production event trigger if the Value of this tag using the Check Sampling Type is less than the defined Check Value.  |
| Check Sampling Type (Optional) | The Check Tag, Check Value, and Check Sampling Type are used to ignore a production event trigger if the Value of this tag using the Check Sampling Type is less than the defined Check Value.  |

## Model 6

Model 6 creates production events when the Batch Tag value transitions from a specific value to another specific value. The Event Number is created using the properties defined for this model. The format of the Event Number is [?YYMMDD##]. The first question mark (?) refers to the first character of the Event Number. The YY refers to a numeric 2 digit year. The MM refers to a numeric 2 digit month. The DD refers to a numeric 2 digit day of the month. The ## refers to the batch or event count of the day.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 6 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Batch Tag                         | The tag used to trigger the creation of a new production event.  |
| From Value                        | The Value the Batch Tag must transition FROM to trigger the creation of a new production event.  |
| To Value                          | The Value the Batch Tag must transition TO in order to trigger the creation of a new production event.   |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Tappi Begin                       | This value is used as the first characters of the Tappi or Event Number.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will  |

|                         |   |
|-------------------------|---|
|                         | impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day. |
| Common PU_Id (Optional) | If a Common PU_Id is supplied, then the Tappi or production event is also created on this common production unit.                           |

## Model 7

Model 7 creates production events when the Batch Tag has a value change and the new value equals the value defined. The Tappi or Event Number is created using the properties defined for this model. The format of the Tappi or Event Number is [?YYMMDD##]. The first question mark (?) refers to the first character of the Event Number. The YY refers to a numeric 2-digit year. The MM refers to a numeric 2 digit month. The DD refers to a numeric 2-digit day of the month. The ## refers to the batch or event count of the day.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 7 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Batch Tag                         | A value change of this tag triggers the creation of a new production event.  |
| The Value                         | The Value the Turnup Tag must equal to trigger the creation of a new Tappi or production event.  |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Tappi Begin                       | This value is used as the first characters of the Tappi or Event Number.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |
| Common PU_Id (Optional)           | If a Common PU_Id is supplied, then the Tappi or production event is also created on this common production unit.  |

## Model 8

Model 8 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the properties defined for this model. The format of the Tappi or Event Number is [M?#####]. The M refers to the Tappi Month characters defined. The first question mark (?) refers to the middle characters of the Tappi or Event Number. The ##### refers to the event count and this sequence number comes from the value of the Turnup Tag.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 8 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Tappi Middle                      | This value is used as the middle characters of the Tappi or Event Number.  |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Month Chars                       | The Tappi Month Characters used to generate the M character.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

## Model 9

Model 9 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the properties defined for this model. The format of the Tappi or Event Number is [M?X?#####]. The M refers to the Tappi Month characters defined. The first question mark (?) refers to the middle characters of the Tappi or Event Number. The X is the first character from the Last Good Value of the defined Other Tag. The second question mark (?) refers to the second set of middle characters of the Tappi or Event Number. The ##### refers to the event count and this sequence number comes from the value of the Turnup Tag.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 9 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Other Tag                         | The Last Good Value is read from the Other Tag and the First character from this value is used as the X character in the Tappi or Event Number.  |
| Tappi Middle #1                   | This value is used as the middle characters of the Tappi or Event Number.  |
| Tappi Middle #2                   | This value is used as the middle characters of the Tappi or Event Number.  |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Month Chars                       | The Tappi Month Characters used to generate the M character.   |

|                     |   |
|---------------------|---|
| Use Prod Day (TRUE) | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day. |
|---------------------|---|

## Model 10

Model 10 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the properties defined for this model. The format of the Tappi or Event Number is [?YM?DD#####]. The first question mark (?) refers to the beginning characters of the Tappi or Event Number. The Y refers to a one-digit year. The M refers to the Tappi Month characters defined. The second question mark (?) refers to the middle characters of the Tappi or Event Number. The DD refers to a numeric 2-digit day of the month. The ##### refers to the event count and this sequence number comes from the value of the Turnup Tag.

---

**NOTE:** The functionality in this model is available only if you have purchased the Quality Module.

---

### Model 10 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                   | Description  |
|----------------------------|--|
| Maximum Run Time (Seconds) | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group     | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                 | A value change of this tag is used to trigger a new production event to be created.  |
| Tappi Begin                | This value is used as the middle characters of the Tappi or Event Number.  |
| Tappi Middle               | This value is used as the middle characters of the Tappi or Event Number.  |
| Prod Day (HH:MM)           | The time that the counter is restarted for the number of events made in that day.  |
| Month Chars                | The Tappi Month Characters used to generate the M character.   |
| Use Prod Day (TRUE)        | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |
| Add Minutes (Optional)     | The number of minutes to add to the event time.  |

## Model 11

Model 11 creates production events when the turnup tag has a value change. The TAPPI or event number is created using the properties defined for this model. The format of the TAPPI or event number is [?M?###]. The first question mark (?) refers to the beginning characters of the TAPPI or event number. The M refers to the TAPPI month characters defined. The second question mark (?) refers to the middle characters of the TAPPI or event number. The ### refers to the event count and this sequence number comes from the value of the turnup tag.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 11 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Tappi Begin                       | This value is used as the middle characters of the Tappi or Event Number.  |
| Tappi Middle                      | This value is used as the middle characters of the Tappi or Event Number.  |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Month Chars                       | The Tappi Month Characters used to generate the M character.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

## Model 12

Model 12 creates production events when the turnup tag has a value change. The TAPPI or event number is created using the properties defined for this model. The format of the TAPPI or event number is [?MDD?##]. The first question mark (?) refers to the beginning characters of the TAPPI or event number. The M refers to the TAPPI month characters defined. The DD refers to a numeric two-digit day of the month. The second question mark (?) refers to the middle characters of the TAPPI or event number. The ## refers to the event count and this sequence number comes from the value of the turnup tag.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 12 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Tappi Begin                       | This value is used as the middle characters of the Tappi or Event Number.  |
| Tappi Middle                      | This value is used as the middle characters of the Tappi or Event Number.  |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Month Chars                       | The Tappi Month Characters used to generate the M character.   |

|                     |   |
|---------------------|---|
| Use Prod Day (TRUE) | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day. |
|---------------------|---|

## Model 13

Model 13 creates production events when the turnup tag has a value change. The TAPPI or event number is created using the properties defined for this model. The format of the TAPPI or event number is [?YM#####]. The first question mark (?) refers to the beginning characters of the TAPPI or event number. The Y refers to a 1 year digit. The M refers to the TAPPI month characters defined. The ##### refers to the event count and this sequence number comes from the value of the Turnup tag.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 13 Properties

The following model properties are set up using the Plant Applications Administrator:

| Property                   | Description  |
|----------------------------|--|
| Maximum Run Time (Seconds) | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group     | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                 | A value change of this tag is used to trigger a new production event to be created.  |
| TAPPI Begin                | This value is used as the middle characters of the TAPPI or Event Number.  |
| Prod Day (HH:MM)           | The time that the counter is restarted for the number of events made in that day.  |
| Month Chars                | The Tappi Month Characters used to generate the M character.   |
| Use Prod Day (TRUE)        | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

## Model 14

Model 14 creates production events when the turnup tag has a value change. The event number is created using the properties defined for this model. The format of the event number is MMM YYYY. The MMM is the three-character month of the month previous to the trigger tag time. The YYYY refers to the a one-digit year. For example, if the trigger tag fires on July 23, 2010, the event name would be JUN 2010. The value for this tag should only change once per month and therefore the event number will be unique.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 14 Properties

The following model properties are set up using the Plant Applications Administrator:

| Property | Description |
|----------|-------------|
|----------|-------------|



|                                   |  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |

## Model 15

Model 15 creates production events when the turnup tag value transitions from a specific value to another specific value. The event number is created using the properties defined for this model. The format of the event number is [?M?###]. The first question mark (?) refers to the 1st character of the event number. The M refers to the TAPPI month characters defined. The second question mark (?) refers to the middle characters of the TAPPI or event number. The ### refers to the batch or event count of the month.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 15 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | The tag used to trigger the creation of a new production event.  |
| From Value                        | The Value the Batch Tag must transition FROM to trigger the creation of a new production event.  |
| To Value                          | The Value the Batch Tag must transition TO in order to trigger the creation of a new production event.   |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number.   |
| Tappi Middle                      | This value is used as the middle characters of the Tappi or Event Number.  |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Month Chars                       | The Tappi Month Characters used to generate the M character.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

## Model 16

Model 16 creates production events when the turnup tag has a value change. The TAPPI or event number is created using the properties defined for this model. The format of the event number is [YY-?MMDD###]. The YY refers to a two-digit year. The first question mark (?) refers to the first characters after the dash. The MM refers to a two-digit month. The DD refers to a two-digit day of the month. The ### refers to the event count of the day.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 16 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                   | Description  |
|----------------------------|--|
| Maximum Run Time (Seconds) | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group     | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                 | A value change of this tag is used to trigger a new production event to be created.  |
| Tappi Begin                | This value is used as the 1st characters of the Tappi or Event Number.   |
| Prod Day (HH:MM)           | The time that the counter is restarted for the number of events made in that day.  |
| Use Prod Day (TRUE)        | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

### Model 17

Model 17 creates production events after every interval elapses. The timing interval and timing offset are parameters defined for this model. For example, if the timing interval was 60 and the offset was 0 then events would be created on the hour. The TAPPI or event number is created using the properties defined for this model. The format of the event number is [?YYMMDD##]. The first question mark (?) refers to the first characters of the event number. The YY refers to a two-digit year. The MM refers to a two-digit month. The DD refers to a two-digit day of the month. The ## refers to the event count of the day.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 17 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                   | Description  |
|----------------------------|--|
| Maximum Run Time (Seconds) | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group     | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Timing Interval (Minutes)  | When the timing interval and offset elapses the production event is created.   |
| Timing Offset (Minutes)    | When the timing interval and offset elapses the production event is created.   |
| Tappi Begin                | This value is used as the 1st characters of the Tappi or Event Number.   |
| Prod Day (HH:MM)           | The time that the counter is restarted for the number of events made in that day.  |
| Use Prod Day (TRUE)        | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

## Model 18

Model 18 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the value in the Turnup Tag with a format like [XYMMDD##] and converting it to a format of [?XYMDD##] and using the properties defined in the model. The 1st question mark (?) refers to the 1st characters after the dash. The M refers to the Tappi Month characters defined. The DD refers to a 2-digit day of the month. The ## refers to the event count of the day.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 18 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number.   |
| Month Chars                       | The Tappi Month Characters used to generate the M character.   |

## Model 19

Model 19 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the properties defined for this model. The format of the Event Number is [?YYYYMMDD##]. The 1st question mark (?) refers to the 1st characters after the dash. The YYYY refers to a 4-digit year. The MMM refers to a 3-character day of the month. The DD refers to a 2-digit day of the month. The ## refers to the event count of the day.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 19 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number.   |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

## Model 20

Model 20 creates production events when the Turnup Tag value transitions from a specific value to another specific value. The Event Number is created using the properties defined for this model. The format of the Event Number is [?YDDD###]. The 1st question mark (?) refers to the 1st character of the Event Number. The Y refers to a numeric 1-digit year. The MM refers to a numeric 2-digit month. The DDD refers to a numeric day of the year. The ### refers to the batch or event count of the day. The Weight, Lineal Footage, and Diameter are also acquired for each event/roll for the defined tags and sampling types. The Sampling Type is a numeric value and can be found in the Sampling\_Type table in the SQL database named GBDB. If the diameter is less than the Minimum Diameter defined, then the event status is set to broke. If the dimensions are unavailable in the Historian, the default dimensions are used.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 20 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | The tag used to trigger the creation of a new production event.  |
| From Value                        | The Value the Batch Tag must transition FROM to trigger the creation of a new production event.  |
| To Value                          | The Value the Batch Tag must transition TO in order to trigger the creation of a new production event.   |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Weight Tag                        | The Tag used to acquire the Roll Weight.   |
| Weight Sampling Type              | The Sampling Type used to acquire the Roll Weight. The Sampling Type is a numeric value and can be found in the Sampling_Type table.   |
| Weight Conv Info                  | The conversion factor to multiply the Roll Weight by. The result is stored as Dimension X.   |
| Lineal Ft Tag                     | The Tag used to acquire the Lineal Footage.  |
| Lineal Ft Sampling Type           | The Sampling Type used to acquire the Lineal Footage. The Sampling Type is a numeric value and can be found in the Sampling_Type table.  |
| Lineal Ft Conv Info               | The conversion factor to multiply the Lineal Footage by. The result is stored as Dimension Y.  |
| Diameter Tag                      | The Tag used to acquire the Roll Diameter.   |
| Diameter Sampling Type            | The Sampling Type used to acquire the Roll Diameter. The Sampling Type is a numeric value and can be found in the Sampling_Type table.   |
| Diameter Conv Info                | The conversion factor to multiply the Roll Diameter by. The result is stored as Dimension Z.   |
| Minimum Diameter                  | The Minimum Diameter value must be met or the Roll event is not created.   |
| Default Weight                    | The Default Weight is used if a value is unavailable in the Historian.   |

|                   |  |
|-------------------|--|
| Default Lineal Ft | The Default Lineal Footage is used if a value is unavailable in the Historian. |
| Default Diameter  | The Default Diameter is used if a value is unavailable in the Historian.       |

## Model 21

Model 21 creates production events when the Turnup Tag value transitions from a specific value to another specific value. The Event Number is created using the properties defined for this model. The format of the Event Number is [?YYYYMMMDD##]. The 1st question mark (?) refers to the 1st characters of the Event Number. The YYYY refers to a numeric 4 digit year. The MMM refers to a 3 character month. The DD refers to a numeric 2 digit day of the month. The ## refers to the event count of the day.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 21 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | The tag used to trigger the creation of a new production event.  |
| From Value                        | The Value the Batch Tag must transition FROM to trigger the creation of a new production event.  |
| To Value                          | The Value the Batch Tag must transition TO in order to trigger the creation of a new production event.   |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number.   |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

## Model 22

Model 22 creates production events when the Turnup Tag value transitions from a specific value to another specific value. The Event Number is created using the properties defined for this model. The format of the Event Number is [?MDD##]. The 1st question mark (?) refers to the 1st characters of the Event Number. The M refers to the Tappi Month characters defined. The DD refers to a numeric 2-digit day of the month. The ## refers to the event count of the day.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 22 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description   |
|-----------------------------------|---|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, |

|                        |   |
|------------------------|---|
|                        | you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop).  |
| Model Processing Group | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.   |
| Turnup Tag             | The tag used to trigger the creation of a new production event.   |
| From Value             | The Value the Batch Tag must transition FROM to trigger the creation of a new production event.   |
| To Value               | The Value the Batch Tag must transition TO in order to trigger the creation of a new production event.  |
| Prod Day (HH:MM)       | The time that the counter is restarted for the number of events made in that day.   |
| Tappi Begin            | This value is used as the 1st characters of the Tappi or Event Number.  |
| Month Chars            | The Tappi Month Characters used to generate the M character.  |
| Use Prod Day (TRUE)    | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day. |

## Model 23

Model 23 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the value in the Turnup Tag with a format like [XXXXXX - Uses Extended API] and converting it to a format of [?XXXXXX] and using the properties defined in the model. The 1st question mark (?) refers to the 1st characters of the event number.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 23 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number.   |

## Model 24

Model 24 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the properties defined for this model. The format of the Event Number is [?MDD##]. The 1st question mark (?) refers to the 1st characters of the Event Number. The M refers to the Tappi Month characters defined. The DD refers to a 2-digit day of the month. The ## refers to the event count of the day.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 24 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| NumSetsTag                        | This value is used to indicate the Number of Sets to create.   |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Month Chars                       | The Tappi Month Characters used to generate the M character.   |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |
| Sets PU_Id (Optional)             | If sets events are to be created this is the PU_Id to create the new set events on.  |
| Sets Lookup Offset (Seconds)      |  |

## Model 25

Model 25 creates production events when the Turnup Tag value transitions from a specific value to another specific value. The format of the Event Number is [XXX###]. The Event Number is created using the first 3 characters (XXX) of the previous event and adding one to the event count that is contained in the last 3 characters.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 25 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | The tag used to trigger the creation of a new production event.  |
| From Value                        | The Value the Batch Tag must transition FROM to trigger the creation of a new production event.  |
| To Value                          | The Value the Batch Tag must transition TO in order to trigger the creation of a new production event.   |

## Model 26

Model 26 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is the value of the Turnup Tag and the Tappi Begin parameter.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 26 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number.   |

### Model 27

Model 27 creates production events when the Turnup Tag value transitions from a specific value to another specific value. The Event Number is created using the properties defined for this model. The format of the Event Number is [YY-?M?####]. The YY refers to a numeric 2 digit year. The 1st question mark (?) refers to the characters after the dash. The M refers to the Tappi Month characters defined. The 2nd question mark (?) refers to the characters after the M. The #### refers to the event count of the day. Set Events are optionally created based on the value in the SetsTag and the Sets PU\_Id.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 27 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | The tag used to trigger the creation of a new production event.  |
| SetsTag                           | This value is used to indicate the Number of Sets to create.   |
| From Value                        | The Value the Batch Tag must transition FROM to trigger the creation of a new production event.  |
| To Value                          | The Value the Batch Tag must transition TO in order to trigger the creation of a new production event.   |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number.   |
| Tappi Middle                      | This value is used as the characters after the M of the Tappi or Event Number.   |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Month Chars                       | The Tappi Month Characters used to generate the M character.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact   |



## Models

|                              |  |
|------------------------------|--|
|                              | when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day. |
| Sets PU_Id (Optional)        | If sets events are to be created this is the PU_Id to create the new set events on.  |
| Sets Lookup Offset (Seconds) |  |

### Model 28

Model 28 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the properties defined for this model. The format of the Event Number is [?YMDDhhmm]. The 1st question mark (?) refers to the 1st characters of the Event Number. The Y refers to a 1 digit year. The M refers to the Tappi Month characters defined. The DD refers to a 2 digit day of the month. The hhmm refers to the hour and minute of the day.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

#### Model 28 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number.   |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Month Chars                       | The Tappi Month Characters used to generate the M character.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

### Model 29

Model 29 calls a stored procedure on a database in the server defined by the ODBC ConnectString property at a specific interval (specified by the TINT property). Each row returned from the stored procedure named in the SpName (NewEvents) property is created as a new production event and, if named, link this new event to another production event. These rows have to be in a specific format. The row ID of each row that is successfully processed is then used as a parameter into the stored procedure named in the SpName (RemoveEvents) property. Use this model over Model 30 or 31 if you need to send customer/shipment information and/or link the event to be created with another event.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

**IMPORTANT:** The ODB connection must NOT have the parameter "Enable Lazy Close Support" selected to avoid locking the Plant Applications tables.

---

#### Model 29 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description   |
|-----------------------------------|---|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop).      |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.   |
| TINT:(Interval) Minutes           | Interval at which the SpName (NewEvents) stored procedure is called on the server named in the ODBCConnectionString.  |
| ODBC Connect String               | DSN=xx;UID=yyy; PWD=zzz;  |
| Machine                           | This is a unique production unit number – not necessarily the PU_Id but just a unique number in the system. The value of this property is passed into the stored procedure named in the NewEvents. This property allows you use the same server and stored procedure for multiple events in Plant Applications. |
| SpName (NewEvents)                | Stored Procedure used to create new events. The header prototype for this stored procedure must look like:<br><br><pre>CREATE PROCEDURE spLocal_NewEvents29 @machnum varchar(30), -- comes from the Machine property @misc varchar(30) -- comes from the Misc property AS</pre>                                 |
| SpName (RemoveEvents)             | Stored Procedure used to remove events. The header prototype for this stored procedure must look like:<br><br><pre>CREATE PROCEDURE spLocal_EventProcessed @ UniqueRowID int --from result set from NewEvents stored procedure AS</pre>   |
| Misc                              | The value of this property will be sent in to the stored procedure defined in the NewEvents property.   |

The result set of this stored procedure must look like this:

```
Declare @ResultSet29 Table(
    UniqueRowID int,
    PlantAppsEventNumber varcahr(25),
    TimestampYear int,
    TimestampMonth int,
    TimestampDay int,
    TimestampHour int,
    TimestampMinute int,
    TimestampSecond int,
    CustomerName varchar(50), --Leave Null if not using customer/shipment
information
    CustomerOrderNumber varchar(25), --Leave Null if not using
customer/shipment information
    OrderStatus varchar(10), --Leave Null if not using customer/shipment
information
    DimX int, --Leave Null if not using customer/shipment information
    DimA int, --Leave Null if not using customer/shipment information
    DimY int, --Leave Null if not using customer/shipment information
```

## Models

```

    DimZ int, --Leave Null if not using customer/shipment information
    ProdCode varchar(25), --Leave Null if not using customer/shipment
information
    LotNumber varchar(10), --Leave Null if not using customer/shipment
information
    ShippedYear int, --Leave Null if not using customer/shipment
information
    ShippedMonth int, --Leave Null if not using customer/shipment
information
    ShippedDay int, --Leave Null if not using customer/shipment information
    ShippedHour int, --Leave Null if not using customer/shipment
information
    ShippedMinute int, --Leave Null if not using customer/shipment
information
    ShippedSecond int, --Leave Null if not using customer/shipment
information
    ParentPlantAppsEventNum varchar(25) --Leave NULL if not linking to
parent production event)
Select * from @ResultSet29

```

## Model 30

Model 30 calls a stored procedure on a database in the server defined by the ODBC ConnectString property at a specific interval (specified by the TINT property). Each row returned from the stored procedure named in the SpName (NewEvents) property is created as a new production event. The timestamp comes from the Turnup Tag. There must be a value in the Turnup Tag within the defined Maximum TT Differential property for the Turnup Tag timestamp to be used as the Event Timestamp. These rows have to be in a specific format. The row ID of each row that is successfully processed is then used as a parameter into the stored procedure named in the SpName (RemoveEvents) property.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

**IMPORTANT:** *The ODB connection must NOT have the parameter "Enable Lazy Close Support" selected to avoid locking the Plant Applications tables.*

---

### Model 30 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | The timestamp of the last good value of this tag is used for the production event timestamp.   |
| TINT:(Interval) Minutes           | Interval at which the SpName (NewEvents) stored procedure is called on the server named in the ODBCConnectString.  |
| ODBC Connect String               | DSN=xx;UID=yyy; PWD=zzz;   |
| Machine                           | This is a unique production unit number – not necessarily the PU_Id but just a unique number in the system. The value of this property is passed in to the stored procedure named in the NewEvents. This   |

|                       |   |
|-----------------------|---|
|                       | property allows you use the same server and stored procedure for multiple events in Plant Applications.   |
| Tappi Begin           | This value is used as the 1st characters of the Tappi or Event Number and is passed into the NewEvents stored procedure.  |
| SpName (NewEvents)    | <p>Stored Procedure used to create new production events. The header prototype for this stored procedure must look like this:</p> <pre>CREATE PROCEDURE spLocal_NewEvents @ TappiBegin varchar(30), --value of the property TappiBegin @machnum varchar(30), --value of the property Machine @misc varchar(30) -- value of the property Misc AS</pre> |
| SpName (RemoveEvent)  | <p>Stored Procedure used to mark the events on the remote system as processed. The header prototype for this stored procedure must look like this:</p> <pre>CREATE PROCEDURE spLocal_EventProcessed @ UniqueRowID int --from result set from NewEvents stored procedure AS</pre>  |
| Max TT Diff (Minutes) | There must be a value in the Turnup Tag within this value for the timestamp of the Turnup Tag to be used as the Event Timestamp.  |
| Log Only (TRUE)       | If True, no events will be created.   |
| Misc                  | Sent into the NewEvents stored procedure.   |
| Var Desc (Optional)   |   |

The result set of this stored procedure must look like this:

```
Declare ResultsSet30 Table(
    UniqueRowId int,
    PlantApplicationsEventNum varchar(25),
    TimestampYear int,
    TimestampMonth int,
    TimestampDay int,
    TimestampHour int,
    TimestampMinute int,
    TimestampSecond int,
    TransType int DEFAULT(1), --means "r;Insert"
    ProdCode varchar(25) DEFAULT('') )-- this value is ignored
Select * from ResultSet30
```

## Model 31

Model 31 calls a stored procedure on a database in the server defined by the ODBC ConnectString property at a specific interval (specified by the TINT property). Each row returned from the stored procedure named in the SpName (NewEvents) property is created as a new production event. These rows have to be in a specific format. The row ID of each row that is successfully processed is then used as a parameter into the stored procedure named in the SpName (RemoveEvents) property.

## Models

Product changes can be automatically created if a new production event has different grade than the previous production event by turning on the Grade Changes parameter.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

**IMPORTANT:** *The ODB connection must NOT have the parameter "Enable Lazy Close Support" selected to avoid locking the Plant Applications tables.*

---

### Model 31 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description   |
|-----------------------------------|---|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop).  |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.   |
| TINT:(Interval) Minutes           | Interval at which the SpName (NewEvents) stored procedure is called on the server named in the ODBCConnectionString   |
| ODBC Connect String               | DSN=xx;UID=yyy; PWD=zzz;  |
| Machine                           | This is a unique production unit number – not necessarily the PU_Id but just a unique number in the system. The value of this property is passed into the stored procedure named in the NewEvents. This property allows you use the same server and stored procedure for multiple events in Plant Applications.                                       |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number and is passed into the NewEvents stored procedure.  |
| SpName (NewEvents)                | <p>Stored Procedure used to create new production events. The header prototype for this stored procedure must look like this:</p> <pre>CREATE PROCEDURE spLocal_NewEvents @ TappiBegin varchar(30), --value of the property TappiBegin @machnum varchar(30), --value of the property Machine @misc varchar(30) -- value of the property Misc AS</pre> |
| SpName (RemoveEvent)              | <p>Stored Procedure used to mark the events on the remote system as processed. The header prototype for this stored procedure must look like this:</p> <pre>CREATE PROCEDURE spLocal_EventProcessed @ UniqueRowID int --from result set from NewEvents stored procedure AS</pre>  |
| Log Only (TRUE)                   | Set to TRUE if you just want to test the stored procedure calls.  |
| Misc                              | The value of this property is passed into the NewEvents stored procedure.   |
| Grade Changes (TRUE)              | Set to TRUE to have the product changed if the new event is a different product than the previous one. Set to FALSE to ignore product changes.  |

The result set of this stored procedure must look like this:

```

Declare ResultsSet31 Table(
  UniqueRowId int,
  PlantApplicationsEventNum varchar(50),
  TimestampYear int,
  TimestampMonth int,
  TimestampDay int,
  TimestampHour int,
  TimestampMinute int,
  TimestampSecond int,
  TransType int DEFAULT(1), --means "r;Insert"
  ProdCode varchar(25) DEFAULT(''), -- Set this to do a product change
  SourceEventNum varchar(50) )
Select * from ResultSet31

```

## Model 32

Model 32 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the properties defined for this model. The format of the Tappi or Event Number is [?M?DD##]. The 1st question mark (?) refers to the beginning characters of the Tappi or Event Number. The M refers to the Tappi Month characters defined. The 2nd question mark (?) refers to the middle characters of the Tappi or Event Number. The DD refers to a numeric 2-digit day of the month. The ## refers to the event count and this sequence number comes from the value of the Turnup Tag.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 32 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Tappi Begin                       | This value is used as the middle characters of the Tappi or Event Number.  |
| Tappi Middle                      | This value is used as the middle characters of the Tappi or Event Number.  |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Month Chars                       | The Tappi Month Characters used to generate the M character.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

### Model 33

## Models

Model 33 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the properties defined for this model. The format of the Tappi or Event Number is [?YYMMDD\_###]. The 1st question mark (?) refers to the beginning characters of the Tappi or Event Number. The YY refers to a 2-digit year. The MM refers to numeric 2-digit month characters. The DD refers to a numeric 2-digit day of the month. The ## refers to the event count and this sequence number comes from the value of the Turnup Tag.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 33 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Tappi Begin                       | This value is used as the middle characters of the Tappi or Event Number.  |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

### Model 34

Set Events are created whenever a new event is manually created on the Production Unit Model 34 is set up on. The Set Events are based on the value in the SetsTag and the Sets PU\_Id. The SetsTag indicates the number of sets to create and the Sets PU\_Id indicates the Production Unit to create the events on.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 34 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| SetsTag                           | This value is used to indicate the Number of Sets to create.   |
| Sets PU_Id                        | If sets events are to be created this is the PU_Id to create the new set events on.  |

### Model 35

Model 35 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the properties defined for this model. The format of the Event Number is [?YMDD##]. The 1st question mark (?) refers to the characters after the dash. The Y refers to a numeric 1-digit year. The M refers to the Tappi Month characters defined. The DD refers to a numeric 2-digit day of the month. The ## refers to the event count of the day.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 35 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Tappi Begin                       | This value is used as the 1 <sup>st</sup> characters of the Tappi or Event Number.   |
| Month Chars                       | The Tappi Month Characters used to generate the M character.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

### Model 36

Model 36 creates production events for Sets when the StartOfSet Tag value transitions from a specific value to another specific value.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 36 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| StartofSet Tag                    | A value change of this tag is used to trigger a new production event to be created.  |
| From Value                        | The Value the Batch Tag must transition FROM to trigger the creation of a new production event.  |
| To Value                          | The Value the Batch Tag must transition TO in order to trigger the creation of a new production event.   |
| Parent PU_Id                      |  |

### Model 37



## Models

Model 37 creates production events for Sets when the StartOfSet Tag value transitions from a specific value to another specific value. The Parent PU\_Id indicates ?. Transition [XXXXXXX#] - End Of Set.

The Length and Diameter are also acquired for each roll for the defined tags and sampling types. The Sampling Type is a numeric value and can be found in the Sampling\_Type table in the SQL database named GBDB.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 37 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| EndOfSet Tag                      | The tag used to trigger the creation of a new production event.  |
| From Value                        | The Value the Batch Tag must transition FROM to trigger the creation of a new production event.  |
| To Value                          | The Value the Batch Tag must transition TO in order to trigger the creation of a new production event.   |
| StartSet PU_Id                    |  |
| Length Sampling Type              | The Sampling Type used to acquire the Length. The Sampling Type is a numeric value and can be found in the Sampling_Type table.  |
| Length Input Tag                  | The Tag used to acquire the Input Length.  |
| Length Tag                        | The Tag used to acquire the Length.  |
| Diameter Sampling Type            | The Sampling Type used to acquire the Roll Diameter. The Sampling Type is a numeric value and can be found in the Sampling_Type table.   |
| Diameter Input Tag                | The Tag used to acquire the Roll Input Diameter.   |
| Diameter Tag                      | The Tag used to acquire the Roll Diameter.   |
| Rolls PU_Id                       | The PU_Id used for the creation of the Rolls.  |

### Model 38

Model 38 creates roll production events when a Set is created. The Minimum Slitter Position Value is used if a value is unavailable in the Historian.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 38 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Rolls PU_Id                       | The tag used to trigger the creation of a new production event.  |
| Slitter Sampling Type             | The Sampling Type used to acquire the Slitter Position. The Sampling Type is a numeric value and can be found in the Sampling_Type table.  |

|                        |  |
|------------------------|--|
| Min Slitter Pos        | The Minimum Slitter Position Value is used if a value is unavailable in the Historian. |
| Min Slitter Difference |  |
| Roll Width Input Tag   | The tag used to acquire the Roll Width.  |
| PI Tag(Optional)       |  |
| PI Tag(Optional)       |  |
| PI Tag(Optional)       |  |
| PI Tag(Optional)       |  |
| PI Tag(Optional)       |  |

## Model 39

Model 39 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the properties defined for this model. The format of the Event Number is [XX-XXXX-###]. The Event Number is created using the first 8 characters (XX-XXXX-) of the previous event and adding one to the event count that is contained in the last 3 characters.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 39 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |

## Model 40

Model 40 creates production events (reels) and grade changes after every Interval is elapsed. The data is retrieved from an external database using an ODBC datasource.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

**IMPORTANT:** The ODB connection must NOT have the parameter "Enable Lazy Close Support" selected to avoid locking the Plant Applications tables.

---

### Model 40 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:(Interval) Minutes           | A value change of this tag is used to trigger a new production event to be created.  |
| ODBC Connect String               | DSN=xx;UID=yyy; PWD=zzz;   |

## Models

|                             |   |
|-----------------------------|---|
| Turnup Time Add (Minutes)   |   |
| Prod Day (HH:MM)            | The time that the counter is restarted for the number of events made in that day. |
| Consumed Char               |   |
| Tappi Addition For Consumed |   |

### Model 41

Model 41 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the properties defined for this model. The format of the Event Number is [?YYMMDD###]. The 1st question mark (?) refers to the 1st characters of the Event Number. The YY refers to a 2-digit year. The MM refers to a 2-character day of the month. The DD refers to a 2 digit day of the month. The ### refers to the event count of the day.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

#### Model 41 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

### Model 42

Model 42 creates production events when the Turnup Tag has a value change. The Event Number is created using the properties defined for this model. The format of the Event Number is [XXXXXXXXXX]. Each X value is retrieved from the 10 Historian Tags defined in the properties.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

#### Model 42 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |

|            |   |
|------------|---|
| Turnup Tag | A value change of this tag is used to trigger a new production event to be created. |
| Tag#1      | The Last Good Value from this tag is used as Character #1.                          |
| Tag#2      | The Last Good Value from this tag is used as Character #2.                          |
| Tag#3      | The Last Good Value from this tag is used as Character #3.                          |
| Tag#4      | The Last Good Value from this tag is used as Character #4.                          |
| Tag#5      | The Last Good Value from this tag is used as Character #5.                          |
| Tag#6      | The Last Good Value from this tag is used as Character #6.                          |
| Tag#7      | The Last Good Value from this tag is used as Character #7.                          |
| Tag#8      | The Last Good Value from this tag is used as Character #8.                          |
| Tag#9      | The Last Good Value from this tag is used as Character #9.                          |
| Tag#10     | The Last Good Value from this tag is used as Character #10.                         |

## Model 43

Model 43 replicates production events to a 2nd Production Unit. The 2nd Production Unit is termed the Slave Unit parameter.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 43 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Slave PU_Id                       | The second Production Unit to replicate events to from this unit.  |

## Model 44

Model 44 creates production events when the turnup tag has a value change. The TAPPI or event number is created using the properties defined for this model. The format of the event number is [?MDD##]. The 1st question mark (?) refers to the first characters of the event number. The M refers to the TAPPI month characters defined. The DD refers to a two-digit day of the month. The ## refers to the event count of the day.

*The functionality in this model is available only if you have purchased the Quality Module.*

### Model 44 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |

|                             |   |
|-----------------------------|---|
| Turnup Tag                  | A value change of this tag is used to trigger a new production event to be created.   |
| NumSetsTag                  | This value is used to indicate the Number of Sets to create.  |
| Prod Day (HH:MM)            | The time that the counter is restarted for the number of events made in that day.   |
| Month Chars                 | The Tappi Month Characters used to generate the M character.  |
| Tappi Begin                 | This value is used as the 1st characters of the Tappi or Event Number.  |
| Use Prod Day (TRUE)         | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day. |
| Sets PU_Id (Optional)       | If sets events are to be created this is the PU_Id to create the new set events on.   |
| Minimum Time Span (Seconds) |   |

## Model 45

Model 45 creates production events when the turnup tag has a value change. The TAPPI or event number is created using the properties defined for this model. The format of the event number is [?Tag2\_DDMMMYY?\_###]. The first question mark (?) refers to the 1st characters of the event number. The Tag2 value is taken from the historian Tag2 defined in the properties. The DD refers to a two-digit day of the month. The MMM refers to a three-character month. The YY refers to a two-digit year. The ## refers to the event count of the day.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 45 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Tag2                              | The value of Tag2 is used in the Event Number.   |
| First Part (?)                    | The 1st question mark (?) refers to the 1st characters of the Event Number.  |
| Second Part (?)                   | The 2nd question mark (?) refers to the characters after the YY.   |

## Model 46

Model 46 creates production events when the turnup tag transitions from a specific value to another specific value. The TAPPI or event number is created using the properties defined for this model. The format of the event number is [?Tag2\_DDMMMYY?\_###]. The first question mark (?) refers to the first characters of the event number. The Tag2 value is taken from the historian Tag2 defined in the properties. The DD refers to a two-digit day of the month. The MMM refers to a three-character month. The YY refers to a two-digit year. The second question mark (?) refers to the characters after the YY. The ## refers to the event count of the day.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 46 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| From Value                        | The Value the Turnup Tag must transition FROM to trigger the creation of a new production event.   |
| To Value                          | The Value the Turnup Tag must transition TO in order to trigger the creation of a new production event.  |
| Tag2                              | The value of Tag2 is used in the Event Number.   |
| First Part (?)                    | The 1st question mark (?) refers to the 1 <sup>st</sup> characters of the Event Number.  |
| Second Part (?)                   | The 2nd question mark (?) refers to the characters after the YY.   |

### Model 47

Model 47 creates production events when the Turnup Tag has a value change. The Tappi or Event Number is created using the Last Good Value in the Turnup Tag with a format like [XXXXXX] and converting it to a format of [?XXXXXX] and using the properties defined in the model. The 1st question mark (?) refers to the 1st characters of the Event Number.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 47 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | A value change of this tag is used to trigger a new production event to be created.  |
| Tappi Begin                       | This value is used as the 1 character of the Tappi or Event Number.  |

### Model 48

Model 48 creates production events when the Turnup Tag value transitions from a specific value to another specific value. The Event Number is created using the properties defined for this model. The format of the Event Number is [?MDD?###]. The 1st question mark (?) refers to the 1st characters of the Event Number. The M refers to the Tappi Month characters defined. The DD refers to a numeric 2-digit day of the month. The 2nd question mark (?) refers to the characters after the DD. The ## refers to the event count of the day.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 48 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | The tag used to trigger the creation of a new production event.  |
| From Value                        | The Value the Batch Tag must transition FROM to trigger the creation of a new production event.  |
| To Value                          | The Value the Batch Tag must transition TO in order to trigger the creation of a new production event.   |
| Tappi Begin                       | This value is used as the 1st characters of the Tappi or Event Number.   |
| Tappi Middle                      | This value is used as the 2nd characters of the Tappi or Event Number where the ? mark is positioned.  |
| Prod Day (HH:MM)                  | The time that the counter is restarted for the number of events made in that day.  |
| Month Chars                       | The Tappi Month Characters used to generate the M character.   |
| Use Prod Day (TRUE)               | If set to TRUE, the Prod Day setting is used to determine the end of the production day and therefore the end of the month. This will impact when the Month Character is changed to the new month. If it is set to FALSE, then midnight is used as the end of the production day.                          |

### Model 49

Model 49 creates production events when the Turnup Tag value transitions from a specific value to another specific value. The Event Number is created using the properties defined for this model. The format of the Event Number is [???X###?YDDD]. The 1st three question marks (???) refers to the 1st characters of the Event Number. The X refers to a value retrieved from the Local Stored Procedure defined in the properties. The ### refers to the event count of the day. The 2nd question mark (?) refers to the characters after the count. The DDD refers to a numeric 3 digit day of the year.

---

*The functionality in this model is available only if you have purchased the Quality Module.*

---

### Model 49 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Turnup Tag                        | The tag used to trigger the creation of a new production event.  |
| From Value                        | The Value the Batch Tag must transition FROM to trigger the creation of a new production event.  |
| To Value                          | The Value the Batch Tag must transition TO in order to trigger the creation of a new production event.   |

|              |   |
|--------------|---|
| Local spName | Used to retrieve the X character in the Event Number.   |
| Tappi Begin  | This value is used as the 1st characters of the Tappi or Event Number.                                |
| Tappi Middle | This value is used as the 2nd characters of the Tappi or Event Number where the ? mark is positioned. |

## Configure Model 118 for Batch Analysis

Model 118 is used as a cross reference to map your batch data to the Plant Applications plant model. When Model 118 is activated, it automatically creates a hidden model (model 49000), which reads the records from the Event\_Transactions table and puts them into the appropriate Plant Applications table.

---

**NOTE:** The functionality in this model is available only if you have purchased the Batch Analysis Module.

---

Plant Applications Administrator > Plant Model > Department > Production Line > Production Unit

To configure Model 118:

1. In the **Plant Applications Administrator**, expand **Plant Model**.
2. Browse to the appropriate production unit, right-click and select **Configure Events on <production unit name>**. The **Event Configuration** wizard appears.
3. On the **Configured Models** tab, under **Add Model**, select **Production Event** from the **Model Type** list and click **Add New Model**.
4. On the **General** tab, click **More Models**. The Search dialog box appears.
5. Select **Model 118 Batch Import** and click **Ok**.
6. Edit the following properties:
  - **optional:** In the **Exclusions** box, type the values this model is to ignore and not use to trigger the model.
  - **optional:** In the **Extended Information** box, type additional information that may be useful. This is not used on any reports or displays.
  - **optional:** In the **Maximum Run Time (Seconds)** box, type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop).
  - **Model Processing Group** is not used for this model.
  - **Area:** This is the value in the Area in your Batch Analysis table and gets mapped to the Department level in the Plant Applications plant model.
  - **Cell:** This is the value in the ProcCell in your Batch Analysis table and gets mapped to the Production Line level in the Plant Applications plant model.
  - **Unit:** This is the value in the Unit in your Batch Analysis table and gets mapped to the Production Unit level in the Plant Applications plant model.
  - **Tag Prefix:** This property is reserved for future use.
  - **Last Number:** optional. Enter the number of the record in the Event\_Transactions table where you want to start processing.
  - **Last Time:** optional. Enter the time of the record in the Event\_Transactions table where you want to start processing.

---

**NOTE:** The Last Time property has precedence over the Last Number property.

---

- **Default Product Family:** optional. Specify the product family your batch product will be associated with. If you leave this blank, your batch product will be associated with the default, "Product Family." (Product family ID = 1)



## Models

- **Data Source:** optional. Specify the data source type, which will be used to map various objects (variables, for example) to the batch data. For example, if you have ProductABC in Plant Applications and the same product is called Batch123 in your batch data, you can create a cross reference so that Plant Applications will know that Batch123 is ProductABC and vice versa. The available data sources are determined by the **active** data sources listed in **Global Configuration > Data Source Types**.

7. Click  **Save** and **Yes** to activate the model.


## Model 49000

Model 49000 is automatically created and configured when [Model 118](#) is configured and activated. Model 49000 reads the records from the Event\_Transactions table and puts them into the appropriate Plant Applications table.

## Detecting Production Events

When configuring production events, Model 800 is the default model. Model 800 can be configured to detect production events, which are used to identify and quantify the material that is to be created and tracked.

To configure Model 800:

1. In the  Plant Model, navigate to the production unit where you want to detect production events.
2. Right-click the production unit and click **Configure Events on <production unit>**. The **Event Detection** wizard appears.
3. On the **Configured Models** tab, select **Production Event** from the **Model Type** list. The **Subtype** list appears.
4. Select a sub-type and click **Add New Model**. Event subtypes define the event label (for example, roll or reel) and the dimensions (for example, litres or meters) for recording event measurements.
5. Click the **General** tab, and do the following:
  - a. **optional:** In the **Auto Move End time (minutes) 0= Off** box, enter a value in minutes for the interval at which you want to move the end time while an event is "in process." The realistic limit is 24 hours (1,440 minutes).
  - b. **optional:** In the **Exclusions** box, type the values this model is to ignore and not use to trigger the model. Use a comma to separate values.
  - c. **optional:** In the **Extended Information** box, type additional information that may be useful. This is not used on any reports or displays.
  - d. **optional:** In the **Maximum Run Time (Seconds)** box, type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop).
  - e. **optional:** Type the group number in the **Model Processing Group** box. The Model Processing Group number can be up to six digits.
  - f. **optional:** Select **Auto Create Applied Products** to automatically create an *applied* product, which will be assigned to the default product family. When a product is not within its target specifications, it is possible to do a search for other products that are within those specifications. If a new product is selected, the selected product becomes an applied product.



- g. **optional:** Select **Add Applied Product to Path** to allow the product to be created on all production execution paths that the production unit is part of and is identified as the schedule point. For more information see Defining Execution Paths.
  - h. **optional:** Select a default product family from the **Default Product Family** list. Products that were automatically created will be assigned to the selected product family. For more information, see Product Families.
  - i. **optional:** Select **Append yymmdd to duplicate events**. This option ensures unique event names. If duplicate event names exist, a unique name will be created by appending the timestamp of the tag to the tag value in the format **\_YYMMDDhhmmss**.
  - j. **optional:** Select **Auto Create Statuses** to create production event statuses if the production status does not exist in the database. For more information, see Production Statuses for Events.
6. Click the **Identify Input(s)** tab to identify and select the historian tags used for input.
- a. Under **Define State Logic Example(s)**, select one of the following options to define the event number.
    - **Set Applied Product**
    - **Set Event Dimension**
    - **Set Event Status**
  - b. Select the checkbox next to one or more of the input types. A new row is added for each input type selected.
  - c. If the tag is to be set as a trigger, select the **Trigger** check box. A minimum of one trigger tag is required.
  - d. Click the **Browse** button beside the **Tag** box. The **Tag Search** dialog box appears. Click **Search** (you can enter search criteria prior to clicking the button if necessary). Select the tag and click **OK**.
  - e. From the **Attribute** list, select either Value or Timestamp, depending on whether you want the value or time of the tag to be passed.
  - f. From the **Sampling Type** list, select the type of sampling that is applied to the tag. For more information on sampling types, see Sampling Types.
  - g. In the **Time Offset** box, enter the number of seconds backward from the trigger time that the next value will be retrieved.



For example:

If you have trigger tag A in the list and you also have another tag B that is used as an input in the VB Script, you can specify what the time offset will be for tag B. This would be the amount of time backward from the trigger time that the EventMgr would attempt to get the 'r;Last Good Value' for tag B. So if tag A changed at 9/28/07 7:00, and you specified a 10 second offset for tag B, then the EventMgr would attempt to retrieve the Last Good Value for tag B starting at 9/28/07 6:59:50.

7. Click the **Scripts** tab to edit sample scripts or write script logic.

The following functionality is available on each tab.


- Click . The **Script Builder** dialog box appears where you can edit the existing script or to write script logic.
- In the **Script Builder** dialog box:
  - Click . The **Select Alias** dialog box appears. Select the Alias and click **OK**. The Alias letter appears in the **Script** box.

- Click  **Check Syntax** to ensure you have entered the VB Script correctly.
  - Click the **Define Event Number** tab to define the event number. "Result" is the required keyword. You can select <User Defined> from the list and set EventNum = NULL to prevent the model from firing.
  - Click the **Define State Logic** tab to either add a new record or update the current record. "State" is the required keyword.
  - Click the **Define Product Logic** tab to define the applied product. "ProdCode" is the required keyword.
  - Click the **Define Dimension X Logic** tab to define the dimension X. "DimX" is the required keyword. Dimension X is the primary dimension used to measure production.
  - Click the **Define Event Status** tab to specify the production status. "EventStatus" is the required keyword. For more information, see Production Statuses for Events.
8. **optional:** Click  **Edit Model Properties**. The Unit Production Event Configuration dialog box appears. Do one or more of the following:
- a. On the General tab, do one of the following:
    - Select **Event Uses Both Start and End Timestamps**. This means that the events will have a start and end time. You will notice that the Start\_Time column in the Events table will be populated.

---

**IMPORTANT:** By default, Event Uses Both Start and End Timestamps is selected and is the recommended setting.

---

    - Select **Event Only Uses End Timestamps**. This means that for each event, the timestamp (which is the end time of the event) will be assumed to be the start time of the next event. You will notice that with this option checked, the Start\_Time column in the Events table does not get populated.
    - Select or clear **Chain Start Times**. If this is not checked, changing the end time of an event will not automatically change the next event's start time. If it is checked, then changing the end time of an event will change the next event's start time.
  - b. To configure a Disposition model, click the Disposition Model tab. For more information, see the topic, [Model 1054-Disposition Model](#), in the online help.
- 
- NOTE:** The Unit Conversion tab is currently not used.
- 
9. Click  to activate the model.

## Replix Models

### Model 60

---

*In order to use this model, you must first purchase a license for the Majiq Replix Interface.*

---

Model 60 is a test data import interface model that imports data from a specific file format generated by Majiq.

Every line in the file is converted to upper case as it is read and any extra spaces at the end of a line are deleted. As the model reads the file, it keeps track of what section of the file it is in. It enters section 1 when it finds a line with "MAJIQ, INC." in it, then moves from section 1 to section 2 when it finds a line that contains "RPT# 009." It moves from section 2 to section 3 when it finds a line that contains "-----" (14 dashes). It enters section 4 when it finds a line that contains "SUB TOTAL".

In section 3, lines are processed into rolls as follows:

LeadRoll = Mid(1,8)  
 RollsPerPack = Mid(10,1)  
 RollNum = Mid(12,8)  
 Customer = Mid(22,8)  
 Order = Mid(31,3);  
 SH = Mid(35,2)  
 Width = Mid(38,4)  
 Dia = Mid(43,4)  
 CT = Mid(48,2)  
 Grd = Mid(51,3)  
 Bwt = Mid(55,3)  
 LinFt = Mid(59,5)  
 Weight = Mid(65,4)  
 Status = Mid(70,4)  
 Bay = Mid(75,4)  
 Vessel = Mid(80,7)  
 H = Mid(87,1)  
 L = Mid(89,1)  
 CC = Mid(91,2)  
 MReelNum = Mid(94,7)  
 Shipped Date = Mid(101,15)  
 AlternateNum = Mid(116,16)

At the end of the file, the import is considered successful if:

- The model read section 4 of the file
- Roll records were created
- The roll data was successfully saved.

### Model 60 Properties

The following model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Purge Days                        | # Days of Accepted/Rejected files to keep  |

### Model 61

## Models

Model 61 is a test data import interface model. It imports Order data from a specific file format generated by an external system.

---

*In order to use this model, you must first purchase a license for the Majiq Replix Interface.*

---

### Model 61 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Purge Days                        | # Days of Accepted/Rejected files to keep.   |

### Model 62

Model 62 is a test data import interface model. It imports Roll data from a specific file format generated by an external system.

---

*In order to use this model, you must first purchase a license for the Majiq Replix Interface.*

---

### Model 62 Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Purge Days                        | # Days of Accepted/Rejected files to keep.   |

### Model 63

Model 63 is a test data import interface model. It imports data from a specific file format generated by Majiq.

---

*In order to use this model, you must first purchase a license for the Majiq Replix Interface.*

---

### Model 63 Properties


The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Purge Days                        | # Days of Accepted/Rejected files to keep  |

## Detecting User-Defined Events

Model 802 can be configured to detect user-defined events. A user-defined event is a generic manually recorded event used to document important occurrences related to process operations. User-defined events could be major maintenance items, routine maintenance items, process checks, shift notes, or any other event important to later analysis. For information about creating models, refer to *Creating Models for User-Defined Events*.

To configure Model 802:

1. In the  Plant Model, right-click the production unit where you want to detect events and click **Configure Events on <production unit>**. The **Event Detection** wizard appears.
2. On the **Configured Models** tab, select **User-Defined Event** from the **Model Type** list. The **Subtype** list appears.
3. Select a sub-type and click **Add New Model**. To create a new sub-type, click **Manage Subtypes**. For more information, see *Configuring User-defined Events*
4. Click the **General** tab and do the following:
  - a. **optional:** In the **Maximum Run Time (Seconds)** box, type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop).
  - b. **optional:** Type the group number in the Model Processing Group box. The Model Processing Group number can be up to six digits.
  - c. **optional:** In the **Extended Information** box, type additional information that may be useful. This is not used on any reports or displays.
  - d. **optional:** In the **Exclusions** box, type the values this model is to ignore and not use to trigger the model. Use a comma to separate values.
  - e. **optional:** Select Chain User Defined Event Times to produce a user-defined event where the start time of the current event is the end time of the previous event.
5. Click the **Identify Input(s)** tab to identify and select the historian tags used for input.
  - a. If the tag is to be set as a trigger, select the **Trigger** check box. A minimum of one trigger tag is required.
  - b. Click the **Browse** button beside the **Tag** box. The **Tag Search** dialog box appears. Click **Search** (you can enter search criteria prior to clicking the button if necessary). Select the tag and click **OK**.




- c. From the **Attribute** list, select either Value or Timestamp, depending on whether you want the value or time of the tag to be passed.
- d. From the **Sampling Type** list, select the type of sampling that is applied to the tag. For more information on sampling types, see Sampling Types.
- e. In the **Time Offset** box, enter the number of seconds backward from the trigger time that the next value will be retrieved.


For example:

If you have trigger tag A in the list and you also have another tag B that is used as an input in the VB Script, you can specify what the time offset will be for tag B. This would be the amount of time backward from the trigger time that the EventMgr would attempt to get the 'Last Good Value' for tag B. So if tag A changed at 9/28/07 7:00, and you specified a 10 second offset for tag B, then the EventMgr would attempt to retrieve the Last Good Value for tag B starting at 9/28/07 6:59:50.

6. Click the **Scripts** tab to edit sample scripts or write script logic.

The following functionality is available on each tab.

- a. Click . The **Script Builder** dialog box appears where you can edit the existing script or to write script logic.
- b. In the **Script Builder** dialog box:
  - o Click . The **Select Alias** dialog box appears. Select the Alias and click **OK**. The Alias letter appears in the **Script** box.
  - o Click  **Check Syntax** to ensure you have entered the VB Script correctly.
  - o Click the **Define UDE Name Logic** tab to edit the VB Script. "UDEName" is the required keyword. This script will be used to generate the user-defined event name when the event is triggered. It will support the expression: "UDE Name = Null" which will signify that there is no event to process and will prevent the model from firing.
  - o Click the **Define State Logic** tab to edit the VB Script. "State" is the required keyword. This script determines when to open or close a user-defined event. Possible values for the result are:  
  
OPEN  
  
CLOSE  
  
OPENCLOSE

7. Click  to activate the model.

## Valmet Models

### Model 88

Model 88 is a test data import interface model. It imports test data from Valmet from a specific file format for a specific site.

---

*In order to use this model, you must first purchase a license for Valmet interface.*

---

### Properties

The following Model properties are set up using the Plant Applications Administrator:

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| TINT:Interval (Minutes)           | Timing Interval the model checks for New Import Files.   |
| Import File Location Spec         | Drive and directory for Import Files.  |
| Accepted File Path                | Drive and directory for Successfully Processed Files.  |
| Rejected File Path                | Drive and directory for Rejected Files.  |
| Tappi Middle                      |  |
| Purge Days                        | # Days of Accepted/Rejected files to keep  |
| Delete Failed Lookups (TRUE)      |  |

## Waste Models

### Waste Event

A Waste event occurs when a product is rejected or lost in a processing stream. It is calculated against a location, which is a specific piece of equipment (equipment module) along a production line causing waste, or around which a material balance was performed to calculate waste. A Production Event can be associated with one or more Waste Events. Each Waste Event has a calculated waste amount using a specific unit of measurement. When stored in the database, waste amounts are normalized to a standard engineering unit based on conversion factors established for each waste measurement. Each Waste Event can have a Cause and an Action Reason applied to it. Cause Reasons are what caused the Waste Event. Action Reasons identify any corrective action taken.

Waste Events and Reasons can be captured automatically using control points and pre-configured Waste Models. Additionally, Waste Events and Reasons can be manually entered. Key Performance Indicator calculations can be created along with ad hoc reports to capture key information regarding Waste Events. This information can also be published to the Web using the Web Report Server.

You must have the license for the Efficiency Management module to configure waste event detection.

### Relationships

**Unit:** The major piece of equipment around which Waste is being tracked.

**Type:** Normally the type of model which produced the Downtime detail. (i.e., "Manual," "Automatic Balance," "Automatic Rejects," or "Manual Rejects")

**Production Event:** The batch waste is associated with or was taken from. More than one waste event can be associated with a given Production Event.

**Measure:** The Unit Of Measurement waste was recorded in. When stored in the database, waste amounts are normalized to a standard engineering unit based on conversion factors established for each waste measurement.

**Amount:** The amount of waste recorded in a specific engineering unit.

**Location:** The specific piece of equipment (equipment module) along a production line causing the waste, or around which a material balance was performed to calculate waste.

**Cause Reasons:** The reasons thought to be the cause of a waste event.

**Action Reasons:** Reasons identifying any corrective action taken.



## Waste Models

The Waste module helps reduce production waste. It automatically calculates process losses, and other waste sources and allows you to identify waste by amount and reason.

In addition, you can use the Waste applications interactive reports to analyze waste information by amount, production unit, and reason. Again, you get the best possible information in the best possible form.

The Waste Models are the logic that determines when waste is produced and calculates how much waste has been produced. Waste Detection Models process input signals to determine if waste has occurred, the source of the waste, and the waste fault if it is available.

Waste Models 300-303 are site specific Waste Tracking Models. These are not the standard recommended models to use to create Waste Events in Plant Applications today. They are very site specific in the calculations that are used to track waste. It is recommended to use the Generic Models for Waste Tracking.

With Model 304, you can use VB Script to enter logic that determines whether there is waste, and also the location, fault, type, and measurement of the waste.

Waste Models 5011, 5012, and 5014 use pre-configured stored procedures so that you can get up and running quickly. When a tag triggers a Waste Event, Model 5011 writes the information associated with the Waste Event, such as waste amount or fault, to the Waste Event table. Model 5012 finds the nearest event at a specific timestamp and then connects the waste to that event. Like Model 5012, Model 5014 finds the nearest event at a specific timestamp, and then connects the waste to that event, but it can also be triggered by historical value changes, and will either create new waste records or update existing ones.

### Log Files

All messages created by models are logged in the Plant Applications Event Manager log files typically in the Proficy\Logfiles directory (this location is a customizable site parameter). The log files are named EventMgr-01.Log and EventMgr-01.Shw with the 01 being the version of the log file. In the "Log" file there will be error messages and in the "Show" file EventMgr.Shw there is a summary of all currently configured models and their current configuration.

## Model 300 – Balancing Waste

With Model 300 you can track how much product went in (Input Tag) to the Production Unit and then compare it to how much product came out (Output Tag). The difference is attributed to Waste. Conversion factors, (**Input Conv Info** and **Output Conv Info**) are used to convert the input units and the output units into common values. For example, if your input is measured in **gallons** and your output is measured in **bottles**, you can use the conversion factors to convert both to **cases**.

---

*The functionality in this model is available only if you have purchased the Efficiency Module.*

---

### Model 300 Properties

The following Model properties are set up using the Plant Applications Administrator:


| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |

|   |   |
|---|---|
| PT:InputTag                               | Tag used to select the Historian tag used to retrieve the amount of product going into the Production Unit.   |
| PT:OutputTag                              | Tag used to select the Historian tag used to retrieve the amount of product coming out of the Production Unit.  |
| Input Conv Info                           | Factor used to convert value from the Input Tag into a common value.  |
| Output Conv Info                          | Factor used to convert value from the Output Tag into a common value.   |
| TINT:Interval (Minutes)                   | The value entered, in minutes, determines how often waste is calculated. For example, if you enter a value of <b>60</b> , then every 60 minutes, starting at midnight, waste would be calculated.             |
| TOFF:Offset (Minutes)                     | The value entered determines the number of minutes after midnight that waste is calculated. For example, if you set the offset at <b>5</b> , then five minutes after midnight, the waste would be calculated. |
| Deadband                                  | Determines the amount of waste that is considered "acceptable."   |
| Samp Type                                 | The Sampling Type determines how to acquire the Input and Output Tag Values.  |
| Lag Time (Minutes)                        | Time delay to wait prior to trying to acquire values from the Historian.  |
| Def Typeld                                | This is the default type you defined when configuring Waste.  |
| Def MeasId                                | This is the default measurement you defined when configuring Waste.   |
| Reason #1                                 | You can type one of the Reasons defined in the Reason Tree  |
| Reason #2                                 | You can type one of the Reasons defined in the Reason Tree  |
| Reason #3                                 | You can type one of the Reasons defined in the Reason Tree  |
| Reason #4                                 | You can type one of the Reasons defined in the Reason Tree  |
| Max Rate Per Min (For Increase Samp Type) | If the sampling type, <b>Increase</b> , is used, this is the maximum rate per minute allowed. Data above this rate is discarded.  |
| Reset Value (For Increase Samp Type)      | If the sampling type, <b>Increase</b> , is used, the Reset Value is the value that the counter resets at.   |





## Model 304

Model 304 determines if there is waste, and also the location, fault, type, and measurement of the waste.


To detect waste on a single location:

1. In the  Plant Model, right-click the production unit where you want to detect events and click **Configure Events on <production unit>**. The **Event Detection** wizard appears.
2. On the **Configured Models** tab, select **Waste** from the **Model Type** list and click **Add New Model**. The **General** tabbed page appears along with the **Reason Tree Configuration**, **Identify Input(s)**, and **Scripts** tabbed pages.
3. Click the **General** tab to select the event type and to set the electronic signature requirements.
  - a. Choose the mode by which the waste event will occur.
    - **Time based waste:** Waste event is not tied to an event.
    - **Event based waste (Exact time):** A production event must exist at the exact time of the waste event to create the waste record. Waste record Event\_Id is populated.
    - **Event based waste (Exact time – create timed waste if event not found):** A production event must exist at the exact time of the waste event to create the


waste record. If there is no production event, then a time-based waste record will be created.

- **Event based waste (Next closest event):** The waste record will be linked to the next closest production event on the unit.
  - b. **optional:** In the **Maximum Run Time (Seconds)** box, type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop).
  - c. **optional:** Type the group number in the Model Processing Group box. The Model Processing Group number can be up to six digits.
  - d. **optional:** In the **Extended Information** box, type additional information that may be useful. This is not used on any reports or displays.
  - e. **optional:** In the **Exclusions** box, type the values this model is to ignore and not use to trigger the model. Use a comma to separate values.
  - f. To require an electronic signature, select the level of authorization from the **Esignature Level** list. For more information on electronic signatures, please see [Using Electronic Signatures](#).
  - g. **optional:** Select **Automatically Add Missing Faults** to add faults if the fault from the fault tag does not exist in the Plant Applications database.
4. Click the **Reason Tree Configuration** tab to associate reason trees and faults to the waste event, and create waste measurement conversions.
- a. Click the **Waste Locations** tab to associate reasons to the waste event.
    - Select the production unit. A message box appears asking if the waste event is time-based.
    - Click Yes or No. The type of waste event is automatically entered under the Association Type column. The **Tree Selection** dialog box appears and three buttons are displayed on the toolbar:  **Assign Cause Tree**,  **Assign Action Tree**, and  **Enable Research**.
    - Select a reason tree and click OK. The selected reason tree is listed under the **Cause Reason Tree** column.
    - Click  **Assign Action Tree** to assign an action reason tree. The **Tree Selection** dialog box appears.
    - Select a reason tree and click **OK**. The selected reason tree is listed under the **Action Reason Tree** column.

---

*NOTE: If you need to create a new action or cause reason tree, click  **Manage Trees**. The **Tree Builder** dialog box appears where you can create new reason trees.*

---



- **optional:** Click  **Enable Research**. **Enabled** is displayed under the **Enable Research** column. Enable Research will enable the Research tab in the Sequence of Events display for waste events. Enabling research allows the user to identify a site user as the person responsible for researching the waste event and to set the research as closed or open.
  - b. Click the **Fault Translation** tab to associate a fault to the waste event.
- Click **Add**. A new row is added under **Fault Translation For Detection Model Output**.
- In the **Fault** column, type a fault value. The fault value must match a value that will be returned by an historian tag (if using model 200) or a fault value identified

in a script (if using model 210 or 211). Fault values must be unique on this production unit.

- In the **Fault Name** column, type a name for the new fault. Fault names must be unique on this production unit.
  - Select the production unit from the **Location** list. If the production unit has slave units configured, the production unit and its slave units will be available in the Location list.
  - In the **Reason1** column select a reason from the list. The contents on the list will depend on the reason tree selected on the Cause Reason Trees tab. Depending on how the reason tree is configured, you may select subsequent reasons for **Reason2** through **Reason4**.
- c. Click the **Conversion/Types** tab to add units and conversion factors to the waste events.
- Click **Constant**, type a description in the Name box and type a conversion value in the Value box. The conversion value is used to convert the waste value to the correct unit of measure used for waste events for a specific production unit.


– OR –

Click **Specification**, to assign a specification variable value as the Value entry. This allows for the use of different conversion factors on a product-specific basis for a given name.




- Under **Waste Types**, type a new waste type. Waste types describe the various types of waste that can occur and can be selected or automatically used to define the type of waste that is being measured or calculated for a waste event.
- d. Click the **Reason Shortcuts** tab to create a shortcut for use on the Waste display.
- d. Click  **Insert Input**. A new row is added.
- Under **Shortcut Name**, type the name of the shortcut. This name will be displayed on the right-click menu in the waste display.
  - Under **Time**, type the duration, in minutes, of the waste event. When the shortcut is applied to a waste event in a waste display, this will become the amount of waste for the event.
  - Under **Location**, select the production unit.
  - Under **Reason Level 1 – 4**, select the reason levels to apply to the waste event in a waste display. The available reasons are determined by the cause reason tree selected in step 4a.
5. Click the **Identify Input(s)** tab to select historian tags for waste inputs.
- a. Under **Waste Input Types**, select the checkbox next to one or more of the waste input types. A new row is added for each input type selected.
  - b. If the tag is to be set as a trigger, select the **Trigger** check box. A minimum of one trigger tag is required.
  - c. Click the  **Browse** button (located next to the **Tag** box). The **Tag Search** dialog box appears. Click **Search** (you can enter search criteria prior to clicking the button if necessary). Select the tag and click **OK**.
  - d. From the **Attribute** list, select either Value or Timestamp, depending on whether you want the value or time of the tag to be passed.
  - e. From the **Sampling Type** list, select the type of sampling that is applied to the tag. For more information on sampling types, see Sampling Types.
  - f. In the **Time Offset** box, enter the number of seconds backward from the trigger time that the next value will be retrieved.

For example:


If you have trigger tag A in the list and you also have another tag B that is used as an input in the VB Script, you can specify what the time offset will be for tag B. This would be the amount of time backward from the trigger time that the EventMgr would attempt to get the 'r;Last Good Value' for tag B. So if tag A changed at 9/28/07 7:00, and you specified a 10 second offset for tag B, then the EventMgr would attempt to retrieve the Last Good Value for tag B starting at 9/28/07 6:59:50.

- g. In the **Precision** box, enter the number of decimal places to keep when reading the Historian tag.
  - h. Click the  **Insert Input** button to add a new input.
6. Click the **Scripts** tab to edit sample scripts or write script logic. On each tab, a sample script is provided, which you can use, or you can click to write script logic.

The following functionality is available on each tab.

- Click . The **Script Builder** dialog box appears where you can edit the existing script or to write script logic.
  - In the **Script Builder** dialog box:
    - Click . The **Select Alias** dialog box appears. Select the Alias and click **OK**. The Alias letter appears in the **Script** box.
    - Click  **Check Syntax** to ensure you have entered the VB Script correctly.
  - a. Click the **Waste Logic** tab to define waste amounts. "Amount" is the required keyword. Click [here](#) for an example.
 

---

*Amount = Null* is used to signify that there was no waste event – exit model.
  - b. Click the **Location Logic** tab to define the waste location using VB Script. "Location" is the required keyword. Click [here](#) for an example.
  - c. Click the **Fault Logic** tab to define the fault logic using VB Script. "Fault" is the required keyword. Click [here](#) for an example.
  - d. Click the **Type Logic** tab to define waste type logic using VB Script. "WasteType" is the required keyword. Click [here](#) for an example.
  - e. Click the **Measurement Logic** tab to define unit of measure logic using VB Script. "Measure" is the required keyword. Click [here](#) for an example.
7. Click  to activate the model.

## Model 304 Sample Logic

### VB Script Example (Model 304)

The following provides example inputs along with corresponding VB Script for Model 304.

#### Identify Model Inputs

- FaultTag
- LocTag
- MeasTag
- TypeTag
- AmountTag (alias = A)
- AmountTag (alias = B)

- AmountTag (alias = C)

### Waste Logic

```
If TypeTag = 1 Then
    Amount = A
ElseIf TypeTag = 2 Then
    Amount = B
ElseIf TypeTag = 3 then
    Amount = C
Else
    Amount = Null
End If
```

### Location Logic

```
Location = LocTag
```

### Fault Logic

```
Fault = FaultTag
```

### Type Logic

```
WasteType = TypeTag
```

### Measurement Logic

```
If TypeTag = 1 Then
    Measure = "Lbs"
ElseIf TypeTag = 2 Then
    Measure = "Tons"
ElseIf TypeTag = 3 then
    Measure = "Kg"
End If
```

## Waste Model 304: Waste Occurs on Single Location

With Model 304, you use VB Script to enter logic that determines whether there is waste, and also the location, fault, type, and measurement of the waste. An easy-to-use editor aids in the creation of scripts, in similar manner to Downtime Models 210-212.

**Maximum # of tags:** 500

**Maximum size of script:** 7000 characters

To configure Waste Model 304:

1. In the Server Manager tree, expand the **Plant Model** and locate the production unit that you want to configure a waste event for.
2. Right-click the production unit and choose **Configure Events on <production unit name>**. The **Event Configuration** dialog box appears, with the **Enable Events** tab active.
3. Under **Available Events to Add**, click **Waste**, and then click **Add Event**. The waste event is added to the list under **Events Enabled On This Unit**.

---

*If this is the first waste event on this unit, ensure that the waste event properties are configured. To do this, select the waste event and click the **Event Properties** button.*

---

4. Ensure that the waste event you just added is selected, and then click the **Configure Event Detection Models** tab.
5. Under **Events Enabled On This Unit**, click **Waste**.
6. Under **Available Models to Assign**, click to select Model 304, and then click **Assign Model**.
7. Under **Custom Configurations**, click **Edit**. The **Waste Model** dialog box appears.

---

*Any changes you make in this dialog box must be saved by clicking the **Save** button in the top left corner. At any time before saving, you can click the **Refresh** button in the top left corner to reset all entries to their last-saved states.*

---

8. Click the **General** tab to enter description information:
  - a. In the **Description** box, enter a description for this instance of Model 304.
  - b. For **Type of Waste Model**, choose the mode by which the waste event will occur.
  - c. In the **Comment** box, enter comments relevant to this configuration.
9. Click the **Identify Model Inputs** tab to enter the tags and triggers for the waste event:
  - a. If the tag is to be set as a trigger, select the **Trigger** check box. A minimum of one trigger tag is required.
  - b. Click the **Browse** button beside the **Tag** box. The **Tag Search** dialog box appears. Click **Search** (you can enter search criteria prior to clicking the button if necessary). Select the tag and click **OK**.
  - c. From the **Attribute** list, select either Value or Timestamp, depending on whether you want the value or time of the tag to be passed.
  - d. From the **Sampling Type** list, select the type of sampling that is applied to the tag.
  - e. In the **Time Offset** box, enter the number of seconds backward from the trigger time that the next value will be retrieved.  
 For example:  
 If you have trigger tag A in the list and you also have another tag B that is used as an input in the VB Script, you can specify what the time offset will be for tag B. This would be the amount of time backward from the trigger time that the EventMgr would attempt to get the 'r;Last Good Value' for tag B. So if tag A changed at 2/28/06 7:00, and you specified a 10 second offset for tag B, then the EventMgr would attempt to retrieve the Last Good Value for tag B starting at 2/28/06 6:59:50.
  - f. In the **Precision** box, enter the number of decimal places to keep when reading the Historian tag.
  - g. Click **Add** to add an additional tag, and repeat steps 9a through 9g until all tags are entered.
10. Click the **Waste Logic** tab to define waste amounts using VB Script. "Amount" is the required keyword.
  - a. **Optional:** Click **Insert Input**. The **Select Alias** dialog box appears. Select the Alias and click **OK**. The Alias letter appears in the **Waste Amount Script** box.
  - b. Enter the text of the VB Script directly in the **Waste Amount Script** box. Click [here](#) for an example.
  - c. Click **Check Syntax** to ensure you have entered the VB Script correctly.

---

*Amount = Null is used to signify that there was no waste event – exit model.*

---

11. Click the **Location Logic** tab to define the waste location using VB Script. "Location" is the required keyword.

- a. **Optional:** Click **Insert Location**. The **Select Location** dialog box appears. Select the location and click **OK**. The location appears in the **Location Logic Script** box.
  - b. **Optional:** Click **Insert Input**. The **Select Alias** dialog box appears. Select the Alias and click **OK**. The Alias letter appears in the **Location Logic Script** box.
  - c. Enter the text of the VB Script directly in the **Location Logic Script** box. Click [here](#) for an example.
  - d. Click **Check Syntax** to ensure you have entered the VB Script correctly.
12. Click the **Fault Logic** tab to define the fault logic using VB Script. "Fault" is the required keyword.
- a. From the **Location** list, select a location to apply the script to. A different script can exist for each location.
  - b. **Optional:** Click **Insert Fault**. The **Select Fault** dialog box appears. Select the fault and click **OK**. The fault appears in the script description box.
  - c. **Optional:** Click **Insert Input**. The **Select Alias** dialog box appears. Select the Alias and click **OK**. The Alias letter appears in the script description box.
  - d. Enter the text of the VB Script directly in the script description box. Click [here](#) for an example.
  - e. Click the **Check Syntax** button to ensure you have entered the VB Script correctly.
13. Click the **Type Logic** tab to define waste type logic using VB Script. "WasteType" is the required keyword.
- a. **Optional:** Click **Insert Input**. The **Select Alias** dialog box appears. Select the Alias and click **OK**. The Alias letter appears in the **Waste Type Logic Script** box.
  - b. Enter the text of the VB Script directly in the **Waste Type Logic Script** box. Click [here](#) for an example.
  - c. Click **Check Syntax** to ensure you have entered the VB Script correctly.
14. Click the **Measurement Logic** tab to define unit of measure logic using VB Script. "Measure" is the required keyword.
- a. **Optional:** Click **Insert Input**. The **Select Alias** dialog box appears. Select the Alias and click **OK**. The Alias letter appears in the **Waste Unit of Measure Script** box.
  - b. Enter the text of the VB Script directly in the **Waste Unit of Measure Script** box. Click [here](#) for an example.
  - c. Click **Check Syntax** to ensure you have entered the VB Script correctly.

### VB Script Example (Model 304)

The following provides example inputs along with corresponding VB Script for Model 304.

#### Identify Model Inputs

1. A - Trigger Tag
2. B - Fault Value
3. C - Waste Type Value
4. D - Waste Identifier
5. E - Amount Value 1
6. F - Amount Value 2
7. G - Amount Value 3



## Models

### Waste Logic

```
If D = 1 Then
    Amount = E
ElseIf D = 2 Then
    Amount = F
ElseIf D = 3 then
    Amount = G
Else
    Amount = Null
End If
```

### Location Logic

```
Location = "My Unit 1"
```

### Fault Logic

```
Fault = B
```

### Type Logic

```
WasteType = D
```

### Measurement Logic

```
If D = 1 Then
    Measure = "Lbs"
ElseIf D = 2 Then
    Measure = "Tons"
ElseIf D = 3 then
    Measure = "Kg"
End If
```

## Model 5011 - Waste Model

### Create Waste at Tag Change Time

When a tag triggers a waste event, Model 5011 writes the information associated with the event, such as amount and Fault type, to the Waste Event table.

---

*The functionality in this model is available only if you have purchased the Efficiency Module.*

---

### Waste Model 5011 Properties

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| PT:Trigger Tag                    | Tag used to trigger the calling of the Stored Procedure  |
| Local SP Name                     | spLocal_Sample5011: This is the name of the pre-configured stored procedure used by this Model.  |
| Amount Tag                        | Tag that contains the Waste amount   |

|                        |   |
|------------------------|---|
| Sampling Type - Amount | Determines how the data is collected from the historian |
| Fault Tag              | Tag that contains the Fault type                        |
| Sampling Type - Fault  | Determines how the data is collected from the historian |

### Waste Model 5011 Stored Procedure Parameters

This stored procedure is pre-configured.

**IMPORTANT:** You can edit this stored procedure. However, if you do edit it, you **must rename it**; otherwise, on the next Plant Applications upgrade, the stored procedure will get overwritten and you will lose any changes that you made.

## Model 5012 - Waste Model

### Create Waste (Closest Event to Tag Change Time)

When a tag triggers a Waste Event, Model 5012 finds the closest Event at a specific timestamp, then connects the Waste event to the Event.

The functionality in this model is available only if you have purchased the Efficiency Module.

### Waste Model 5012 Properties

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| PT:Trigger Tag                    | Tag used to trigger the calling of the Stored Procedure  |
| Local SP Name                     | spLocal_Sample5012: This is the name of the pre-configured stored procedure used by this Model.  |
| Amount Tag                        | Tag that contains the Waste amount   |
| Sampling Type - Amount            | Determines how the Amount data is collected from the historian   |
| Fault Tag                         | Tag that contains the Fault type   |
| Sampling Type - Fault             | Determines how the Fault data is collected from the historian  |

### Waste Model 5012 Stored Procedure Parameters

This stored procedure is pre-configured.

**IMPORTANT:** You can edit this stored procedure. However, if you do edit it, you **must rename it**; otherwise, on the next Plant Applications upgrade, the stored procedure will get overwritten and you will lose any changes that you made.

## Model 5014 - Waste Model

### Create Waste (Closest Event to Tag Change Time) from Autolog

When this model is activated on a given unit, the following occurs.

- A new variable group called "Model 5014 Calculation" is created for the unit.
- A new calculation variable is created for the unit, with a name formatted as:

*Unit <unit number> Model 5014 Autolog Waste Calc*

## Models

Where <unit number> is the database identity (PU\_Id) of the unit being modified.

When a tag triggers a waste event, Model 5014 finds the closest event at a specific timestamp, then connects the waste event to the event. It can also be triggered by historical value changes, and will either create new waste records or update existing ones.

### Waste Model 5014 Properties

| Property                          | Description  |
|-----------------------------------|--|
| <b>Maximum Run Time (Seconds)</b> | <b>optional:</b> Type the number of seconds you want the model to run. The default is 0 (zero), which means the model will not time out. Typically, you will want to limit the run time of the model only if troubleshooting the model (for example, one of the stored procedures is in an infinite loop). |
| Model Processing Group            | Used for multithreading. See the <a href="#">Multithreading</a> topic for more information.  |
| Local SP Name                     | spMSICalc_Sample5014Calc: This is the name of the pre-configured stored procedure used by this model   |
| Amount Tag                        | Tag that contains the waste amount   |
| Fault Tag                         | Tag that contains the fault type   |
| Waste Type                        | The type of waste event  |
| Waste Measure                     | The unit of measure for the waste event  |

If necessary, more than one waste model (5014) can be configured for a unit.

### Waste Model 5014 Stored Procedure Parameters

This stored procedure is pre-configured.