Oracle® Agile Product Lifecycle Management for Process System Events Extensibility Guide

Feature Pack 4.2

E66826-01

April 2016

ORACLE®

Copyrights and Trademarks

Agile Product Lifecycle Management for Process

Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface

Audience

This guide is intended for client programmers involved with integrating Oracle Agile Product Lifecycle Management for Process. Information about using Oracle Agile PLM for Process resides in application-specific user guides. Information about administering Oracle Agile PLM for Process resides in the *Oracle Agile Product Lifecycle Management for Process Administrator User Guide*.

Variability of Installations

Descriptions and illustrations of the Agile PLM for Process user interface included in this manual may not match your installation. The user interface of Agile PLM for Process applications and the features included can vary greatly depending on such variables as:

- Which applications your organization has purchased and installed
- Configuration settings that may turn features off or on
- Customization specific to your organization
- Security settings as they apply to the system and your user account

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Software Availability

Oracle Software Delivery Cloud (OSDC) provides the latest copy of the core software. Note the core software does not include all patches and hot fixes. Access OSDC at:

http://edelivery.oracle.com

Chapter 1—Introduction

Purpose

This guide describes how to use the Event Framework to subscribe to, and act upon, system generated events. While certain events already exist in the application for specific auditing purposes, the System Event framework provides a consistent, extensible, and flexible way to handle raised events throughout the application suite.

This document provides:

- A business and technical overview of the Event Framework
- Details on how to configure and create event subscribers
- Details on creating reusable event subscriber filters
- Details of existing out-of-the-box event subscribers and filters
- Information on reference implementations of event subscribers and filters
- Instructions on using the Database Event expiration RemotingContainer service

Overview

The Agile PLM for Process application can optionally raise events, such as Save, Workflow, Read, Print, and others, for high-level business objects, which can be subscribed to and acted upon programmatically. A raised event passes specific contextual information, such as the specification, NPD activity, etc., to any configured subscribers. Additional information, such as status details, and related object information is also included. The subscriber, written as a class (C#), can then act on the given event arguments as need. For instance, the subscriber may send an email notification to an NPD activity owner once a related GSM Specification is approved. Multiple event subscribers can be called for a single event, and a single subscriber can be used for multiple events.

Customers specify the events to subscribe to in the CustomEventing.xml configuration file, which allows for flexible configuration options, such as:

- Subscribing to a trade specification new issue event when the trade specification is not in an Approved status
- Subscribing to any specification type Print events
- Subscribing to a NPD project save event when a project is in Stage 3.

Supported applications

System event extensibility is currently available in the following applications:

- **GSM**—All GSM specifications and specification templates, signature documents, testing protocols, and smart issue events, print events
- **SCRM**—Facility, company, sourcing approval, non-specification related sourcing approval, signature documents, supplier documents, and related template events
- **NPD**—Project and activity events, print events

- **PQM**—PQM action, issue, audit (and template), and signature document events
- **All applications**—Authentication events for each application (successful login, inter-app authentication, single sign on).

Supported events

Raised event types differ by object and application, but will typically include:

- Read
- Edit
- Pre-Save
- Save
- Create
- Create From Template
- Workflow
- Copy
- Authenticate
- … and more
  The full listing of raised events, along with the event argument detail, is listed in the *Event Types and Associated Objects.xlsx* document available in the Feature Pack's ReferenceImplementations\EventFrameworkExtensions\Documentation folder,.

Event Information

Raised events include specific contextual information that may be used by event subscribers, such as the object being acted on, status information of the object, user, timestamp, and more. The following list shows some of the possible data passed to the subscriber:

- App Name (ex: "GSM")
- Event Name (ex: "GSM.Item.5816.Save" for a formulation specification Save event)
- Primary object (ex: Formulation specification data object)
- Secondary object (ex: original item/object used for a copy event)
- User
- Status IDs—List of the item's status identifiers. For specifications, PQM items, sourcing approvals, this is a list of workflow tag IDs (ex: 4 is the tagID for IsApproved). For NPD, uses the related stage numbers, gate into, etc.
- Additional information (three additional info fields)
- Timestamp
- Sender—The source object/service that triggers the event (eg. SpecificationService)

More details can be found in the *Event Types and Associated Objects.xlsx* document.

Chapter 2—Event Subscribers

Event subscribers are classes, written in C#, that handle one or more raised events. When a system event is raised, all configured event subscribers that match the given event are called. You can control when an event subscriber can get called in several ways, such as by the event name, the object status, and more. You can therefore configure a single subscriber to handle multiple events, and can have multiple subscribers for a single event.

> For instance, and event subscriber that subscribes to "GSM.Item.1004.*" will get called for all Material Specification events.

See the Configuration section for more details.

Subscriber classes are called synchronously to the core system transactions. For instance, when a specification is loaded for viewing, the system will load the specification from the database, verify that user can access it, and then raise the Read event (e.g., "GSM.Item.1004.Read" for a material specification). All matching event subscribers will then get called. Only when the called subscribers complete their actions will the core business logic continue and display the specification to the user.

Performance Considerations

The performance impact of calling a subscriber will depend on the specific subscriber implementations. Subscribers should limit their actions to quick actions that will not delay the main applications when possible. If the subscriber class causes some exception, then the core action will not complete, and an error will be experienced by the user. The Performance Instrumentation feature can be turned on to evaluate the impact of individual subscribers.

Existing Subscribers

The following out of the box event subscribers are available for use.

Database Logger Event Subscriber

The Database Logger saves the basic event information in the CommonEventingLog table, recording the various Event argument objects using their PKIDs.

SequenceNum	AppName	EventName	PrimaryObjectID	Secondar...	LogDate	UserID	Comments	StatusIDs	OtherInfo1	OtherInfo2	OtherInfo3
179	PQM	PQM.Item.7002.Save	7002000FFK		2014-10-24 11:01:58.000	205302a693e2-01a...		101	2090f89a2c9d-34ff-4536-...	575817617a4b-c18d-411e-...	20
178	PQM	PQM.Item.7002.PreSave	7002000FFK		2014-10-24 11:01:58.000	205302a693e2-01a...		101	2090f89a2c9d-34ff-4536-...	575817617a4b-c18d-411e-...	20
177	PQM	PQM.Item.7002.Edit	7002000FFK		2014-10-24 11:01:25.000	205302a693e2-01a...		101	2090f89a2c9d-34ff-4536-...	575817617a4b-c18d-411e-...	20
176	PQM	PQM.Item.7002.Read	7002000FFK		2014-10-24 10:55:08.000	205302a693e2-01a...		101	2090f89a2c9d-34ff-4536-...	575817617a4b-c18d-411e-...	20
175	PQM	Authenticate	205302a693e2-01a8-4edd-8...		2014-10-24 10:54:56.000	205302a693e2-01a...			AuthenticationService	ProdikaAuthenticationStrate...	

Note that data saved in the CommonEventingLog table is intended for event-related purposes only and is **not an auditing table**, as it is regularly purged using the DatabaseEventingExpirationService RemotingContainer service. See the Database Eventing Expiration Service section for more details.

Using the Database Logger

To use the Database Logger, the event subscriber configuration can:

1. Use the InheritFromId attribute, referencing the “CommonDatabaseLogger” and leaving the FactoryURL empty. For example:

```
<SubscriberConfig
    Id="exampleSubscriberUsingInherit"
    EventTypes="GSM.Item.5816.Read"
    InheritFromId="CommonDatabaseLogger"
    FactoryURL="" Enable="true"></SubscriberConfig>
```

2. Use the FactoryURL, specifying the class "Class:Xeno.Prodika.Eventing.Subscriber.DatabaseLoggerEventSubscriberFactory,GeneralServices" directly. For example

```
<SubscriberConfig
    Id="exampleSubscriberUsingFactoryURL"
    EventTypes="GSM.Item.5816.Read"
FactoryURL="Class:Xeno.Prodika.Eventing.Subscriber.DatabaseLoggerEventSubscriberFactory,Genera
lServices"
    Enable="true"></SubscriberConfig>
```

Logging to other tables

The Database Logger subscriber is also able to record event information to a *different* table, as long as the table has the same schema as the CommonEventingLog table. For example, if you wanted to log the specification Read and Print events to one table, or PQM events to their own table, you would create a new table just like CommonEventingLog, say PQMEventingLog. Then, the subscriber factory is configured using this table name as a parameter (using a $ sign as a separator). For example:

```
<SubscriberConfig Id="PQMAll" EventTypes="PQM.*" StatusIds=""
FactoryURL="Class:Xeno.Prodika.Eventing.Subscriber.DatabaseLoggerEventSubscriberFactory,Genera
lServices$PQMEventingLog" . . .
```

File Logger Event Subscriber

The File Logger Event Subscriber is a simple rolling file logger that writes the current event arguments to a rolling file.

```
1  2014-10-28 09:06:58 406 - INFO - APP:GSM, EventName:GSM.Item.1004.Read, DataObjectPKID:100451432782-b0ac-401b-b838-072bd6de1856,
   SecondaryObjectPKID:, User:205302a693e2-01a8-4edd-8730-873f7b2bcfef, Comment:, TagIDs: 102,101,
   OtherInfo1:20907bceac7a-ccc3-4e37-afdc-02f33e3d2941, OtherInfo2:5758c26f2ed1-e098-4fb4-98bb-ceclaf89170a, OtherInfo3:
2  2014-10-28 09:09:42 325 - INFO - APP:GSM, EventName:GSM.Item.1004.Read, DataObjectPKID:100451432782-b0ac-401b-b838-072bd6de1856,
   SecondaryObjectPKID:, User:205302a693e2-01a8-4edd-8730-873f7b2bcfef, Comment:, TagIDs: 102,101,
   OtherInfo1:20907bceac7a-ccc3-4e37-afdc-02f33e3d2941, OtherInfo2:5758c26f2ed1-e098-4fb4-98bb-ceclaf89170a, OtherInfo3:
3
```

By default, it will log to the C:/PLM4P/Logs/EventingLogs directory, using the filename Eventing.txt, which will have the date appended to it. However, a different path and filename can be specified.

Using the File Logger

To use the File Logger, the event subscriber configuration can:

1. Use the InheritFromId attribute, referencing the “CommonFileLogger” and leaving the FactoryURL empty. For example:

```
<SubscriberConfig
    Id="exampleSubscriberUsingInherit"
```

```
    EventTypes="GSM.Item.1004.*"
    InheritFromId="CommonFileLogger"
    FactoryURL="" Enable="true"></SubscriberConfig>
```

2. Use the FactoryURL, specifying the class "Class:Xeno.Prodika.Eventing.Subscriber.FileLoggerEventSubscriberFactory,GeneralServices" directly. For example

```
<SubscriberConfig
    Id="exampleSubscriberUsingFactoryURL"
    EventTypes="GSM.Item.1004.*"
FactoryURL="Class:Xeno.Prodika.Eventing.Subscriber.FileLoggerEventSubscriberFactory,GeneralServices"
    Enable="true"></SubscriberConfig>
```

Logging to other files

The File Logger subscriber is also able to record event information to a different directory and filename. For example, if you wanted to log the specification Read and Print events to one file, the subscriber factory is configured using this filename name. For example:

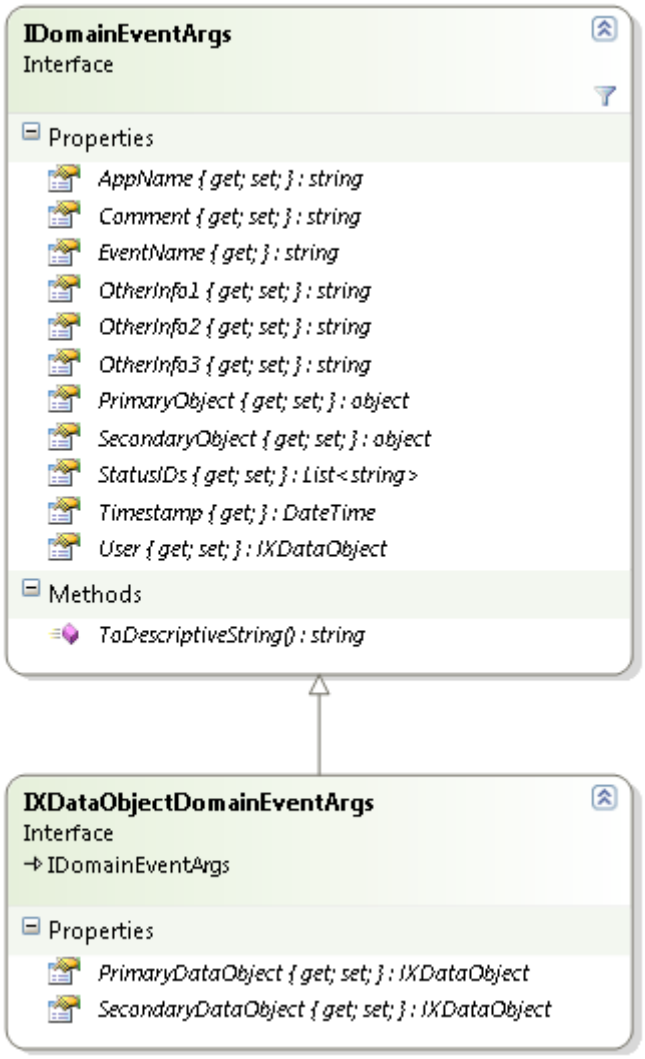
```
<SubscriberConfig Id="PQMAll" EventTypes="GSM.Item.*.Read,GSM.Item.*.Print" StatusIds=""
FactoryURL="Class:Xeno.Prodika.Eventing.Subscriber.FileLoggerEventSubscriberFactory,GeneralServices$C:/PLM4P/Logs/ReadEventingLogs|EventLog_Read" . . .
```

Custom Subscribers

Customer can implement their own custom subscriber classes. Event subscriber classes are created by an Event Subscriber Factory class that must implement the `Xeno.Prodika.Eventing.Subscriber.IEventSubscriberFactory` interface from the GeneralServices.dll assembly.

The factory creates an Event Subscriber via the Create method. The Event Subscriber class must implement the `Xeno.Prodika.Eventing.Subscriber.IEventSubscriber` interface from the GeneralServices.dll assembly.

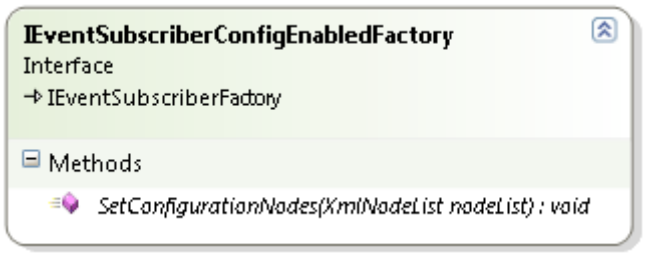
The Event Subscriber class must implement the HandleEvent method, which takes the event sender object and the event arguments (IDomainEventArgs).



The DomainEventArgs are in the Xeno.Prodika.Eventing namespace of GeneralServices.dll, and contain the primary event information.

An additional Property, SegmentExternalIDs, is also available as a List<string> of the segment external ID values.

Some Event Subscriber Factory classes can leverage configuration Filters (see the Configuration section below) by implementing the IEventSubscriberConfigEnabledFactory interface, which has a SetConfigurationNodes method.



This allows the subscriber factory to get the xml defined for its configuration entry, allowing for the use of existing filters or custom filters.

The abstract class AbstractFilteredSubscriberFactory is available to extend your Subscriber Factory from. It will process any XML configuration nodes defined in the subscriber configuration (as Filters) and makes a list of these subscriber filter classes available to the Event Subscriber that your class creates. Read more about Filters in Chapter 3—Configuration.

Reference Event Subscriber

A reference implementation of an event subscriber class is available in the ReferenceImplementations\EventFrameworkExtensions\SourceCode folder.

Allergens Contained Change Notification Subscriber

This reference implementation is an example subscriber class that demonstrates how event subscribers can be written. It works by subscribing to two main events, the Edit and Save events. On Edit, it takes a snapshot of the current specification Allergens Contained listing, and uses that to compare with the values present on the Save event. Values that have changed, been added, or removed, and composed into a simple email message and sent to configured recipients.

See the Reference Implementation source code for more specific examples.

Chapter 3—Configuration

System Events are controlled through configuration.

Feature Configuration

The System Events are ***disabled by default*** but can be turned on using the feature configuration Common.SystemEventing.Enabled entry. To enable System Events, add the following entry to the CustomerSettings.config file, in the FeatureConfiguration node.

```
<add key="Common.SystemEventing.Enabled" value="true"/>
```

Subscriber Configuration

Event subscribers are specified in the CustomEventing.xml configuration file in the config\Extensions directory

The event subscribers are configured by the event name(s), optional status filters, the event subscriber class that should get called for the event, and more. The following is an example of a simple subscriber configuration for a Formulation Spec Copy event:

```
<SubscriberConfig Id="exampleSubscriber"
       EventTypes="GSM.Item.5816.Copy"
       FactoryURL="Class:Xeno.Prodika.Eventing.Subscriber.DatabaseLoggerEventSubscriberFactory
,GeneralServices"
       Enable="true"></SubscriberConfig>
```

The configuration allows for some helpful rules to indicate when the subscriber should be called for a given event of set of events:

Controlling subscribers by event names

- Included Event Names—Subscribers are primarily specified by event names
  - **Exact event names** can be used to subscribe to a specific event, using the Application code, the general item type (Item or Template), the 4 digit object type code, and the event. For instance, a configuration entry attribute of **`EventTypes="GSM.Item.1004.Save"`** would call the subscriber for a material specification save event.
    - Note that some event names are specified differently, such as ‘Authenticate’ or ‘GSM.SmartIssue.EndProcessing’
    - Event names can be found in the Event Types.xlsx document.
  - **Wildcard characters in the event name**. This allows a subscriber to be called for a common set of events. For instance, a configuration entry attribute of **`EventTypes="GSM.Item.*.Read"`** would call the subscriber for any GSM specification Read event, excluding templates , while a configuration entry of **`EventTypes="GSM.Template.*.Create"`** would use the subscriber for any GSM template Create event. A configuration entry of **`EventTypes="GSM.*.*.Workflow"`** would use the subscriber for any workflow event of GSM items (including specification templates).

  - **Comma separated list of events.** Allows for a list of event names to be used. For example, a configuration entry of **`EventTypes="GSM.Item.*.Print,GSM.Item.*.Read"`** would use the subscriber for any specification Print or Read events.
  - **Regex for event types**
- **Excluded Event Names** —This setting allows the subscriber to be skipped if it matches an excluded event types rule. The Excluded listing rules also allow for wildcard characters and comma separated entries. For example, a configuration entry of **`EventTypes="GSM.Item.*"`** **`ExcludedEventTypes="GSM.Item.*.Read"`** would include all GSM specifications (not templates) except for Read events.

Controlling subscribers by status

- **List of status identifiers.** Event subscribers can be included based on the status of the object. A comma separated list of status identifiers can be used to limit the subscriber to only be called if it is in one of the listed status identifiers.
  Status identifiers differ by application.
  - For GSM specifications, PQM items, and SCRM sourcing approvals, the workflow tag behavior IDs are used (For example, 4 is the Is Approved value).
  - For NPD projects, the stage number is used, along with additional rules; a project in Stage 3 would be S3, while a Stage 2 Gate would be S2G.
  - Additional details can be found in the Eventing spreadsheet.
- **List of excluded status identifiers.** Events can also be excluded based on the status of the object. This uses the same format as the included status listing, except it will exclude any items having a matching status.

Controlling subscribers by Filters

To further control when subscribers are called, subscriber Filters can be used in the configuration.

- The existing **UserRole** and **UserGroup** filters allow you to specify User Roles and Groups as subscriber criteria. Included and Excluded Roles and Groups can be configured easily. For example, the following sample config uses both the Group and Role filters…:

```
<SubscriberConfig Id="exampleSubscriber"
       EventTypes="GSM.Item.5816.Copy"
       FactoryURL="Class:Xeno.Prodika.Eventing.Subscriber.DatabaseLoggerEventSubscriberFactory
,GeneralServices"
       Enable="true">
       <Filters>
              <Filter
FactoryURL="Class:Xeno.Prodika.GeneralServices.Eventing.Filters.UserGroupFilter,PlatformExtens
ions">
                     <GroupsIncluded>20558c73dc8d-fe2c-4b28-b4df-e984ecf8d901,205516328111-
50df-4b81-b371-6efdfca70304</GroupsIncluded>
                     <GroupsExcluded>2055179E44BB-BB70-4AF3-8F95-
4C41D66617E4</GroupsExcluded>
              </Filter>
              <Filter
FactoryURL="Class:Xeno.Prodika.GeneralServices.Eventing.Filters.UserRoleFilter,PlatformExtensi
```

```
ons">
                        <RolesIncluded>[SPEC_CREATOR_1004],[SPEC_CREATOR_2147]</RolesIncluded>
                        <RolesExcluded></RolesExcluded>
                </Filter>
        </Filters>
</SubscriberConfig>
```

- An additional filter, **SegmentsFilter**, can be used to include and/or exclude an object based on its assigned segments. Use FactoryURL="Class: Xeno.Prodika.GeneralServices.Eventing.Filters.SegmentsFilter,PlatformExtensions" and the nodes IncludedItems and ExcludedItems:

```
<Filter FactoryURL=
"Class:Xeno.Prodika.GeneralServices.Eventing.Filters.SegmentsFilter,PlatformExtensions">
      <IncludedItems>alpha,gamma</IncludedItems>
      <ExcludedItems>Beta</ExcludedItems>
</Filter>
```

- Another existing filter, **ReflectiveSimplePropertyFilter**, allows for a flexible filter option; in the xml configuration, you can specify an object property to filter on, and the IncludedItems and Excluded Items nodes that should be used to filter it. The object property cannot be a collection, and must use the PropertyPath syntax as described in the *Database and Object Schema Guide*.
    - For instance, this will filter a trade specification event subscriber to only be triggered if the trade item type is TU or TU Co-pack (the ids for the types are 3 and 7).

```
<Filter
FactoryURL="Class:Xeno.Prodika.GeneralServices.Eventing.Filters.ReflectiveSimplePropertyFilter
,PlatformExtensions">
      <PropertyPath>TradeType.TypeID</PropertyPath>
      <IncludedItems>3,7</IncludedItems>
</Filter>
```

    - While this will include every trade item type that is *not* a 'TU no children'

```
<Filter
FactoryURL="Class:Xeno.Prodika.GeneralServices.Eventing.Filters.ReflectiveSimplePropertyFilter
,PlatformExtensions">
      <PropertyPath>TradeType.TypeID</PropertyPath>
      <ExcludedItems>4</ExcludedItems>
</Filter>
```

- **Custom filters** can be created for additional control. The customer filter is a C# class that must implement the `Xeno.Prodika.Eventing.Subscriber.Filters.IEventSubscriberFilter` interface from the GeneralServices.dll assembly. A custom filter class is passed the xml nodes inside the <Filter> element of the config, via the SetConfigNodes method, right after the class is created. The 'Include' method can then be called to determine if the filter allows the current event or

not.

- Many filters are based on a list of items to include and/or exclude. Therefore, your filter can **extend the abstract class AbstractIncludeExcludeEventSubscriberFilter** (using Xeno.Prodika.GeneralServices.Eventing.Filters from PlatformExtensions.dll), which will automatically read a Filter configuration that has the XML nodes <IncludedItems> and <ExcludedItems>. The **IncludedItems** and **ExcludedItems** properties are available to your implementation class as a list of strings, which are read from the configuration (separated by commas). See the SegmentsFilter xml example configuration above. This makes implementing a filter much simpler since you don't have to handle reading the configuration file data.
- **A Reference Implementation** of a subscriber filter is available in the ReferenceImplementations\EventFrameworkExtensions\SourceCode folder. The example code, MaterialClassificationFilter, uses a material specification Classification as a filter, and allows for a list of classifications for inclusion and a list for exclusion. It demonstrates how you can create your own reusable filters, which can be configured for other event subscribers.

Other Subscriber Configuration Attributes

The Id attribute is a required attribute and is used to uniquely identify the subscriber configuration entry. The id can be used to reference this subscriber's FactoryURL setting by another subscriber, via the InheritFromId attribute (see below).

The FactoryURL attribute is used to specify the EventSubscriberFactory class that should be used for the given subscriber. It uses the ObjectURL syntax, as specified in the Extensibility Guide Appendix.

The InheritFromId attribute is a simple convenience function which can be used, if desired, to inherit the FactoryURL value from another subscriber. For example, in the following configuration the SCRMAll subscriber will use the same class that the PQMAll subscriber uses:

```
<SubscriberConfig Id="PQMAll" EventTypes="PQM.*"   StatusIds=""
FactoryURL="Class:Xeno.Prodika.Eventing.Subscriber.DatabaseLoggerEventSubscriberFactory,GeneralServ
ices" InheritFromId="" Enable="" ></SubscriberConfig>
<SubscriberConfig Id="SCRMAll" EventTypes="SCRM.*"   StatusIds=""  InheritFromId="PQMAll" Enable=""
></SubscriberConfig>
```

Note that *only the FactoryURL attribute is inherited*; the other attributes, such as EventTypes, are not.

The Enable attribute can be set to true or blank to enable the subscriber, or false to turn off the subscriber.

Note that any configuration file changes require an IIS Reset to be picked up.

Example subscriber config entry

```
<SubscriberConfig Id="ApprovedGSMLogger" EventTypes="GSM.*" ExcludedEventTypes="GSM.Template.*"
StatusIds="4" ExcludedStatusIds="1022"
FactoryURL="Class:Xeno.Prodika.Eventing.Subscriber.DatabaseLoggerEventSubscriberFactory,GeneralServices"
InheritFromId="" Enable=""></SubscriberConfig>
```

The above subscriber configuration would include all GSM events in an Approved status (StatusIDs=”4”) excluding specifications having a workflow tagID of 1022 and excluding templates. The event would then call the class specified by FactoryURL. In this case, the class is a database logger class which records the event information in the CommonEventingLog table

Chapter 4—Remoting Container Service

When used, the Database Logger event subscriber (described in Chapter 2) will record configured events to a database table. By default, the table used by the DatabaseLogger is the provided table CommonEventingLog. Customers can leverage this table for database triggered actions, if desired. However, depending on the number of events logged to this table, the size of the table may grow quickly. Therefore, the table is not intended to retain event data records permanently, and as such, should not be used for reporting purposes. Customers are encouraged to record the specific events they are interested to their own tables. For instance, a nightly database process can be used to copy certain events to another database that can be used for reports. Customers can also use the Database Logger event subscriber to record events directly to their own tables.

Database Eventing Expiration Service

Because the volume of data added to the table may be large, the **Database Eventing Expiration Service**, hosted on the **Remoting Container**, should be used to regularly purge records from the table.

Configuration

To enable the expiration service, modify the PLM4P.DBEventingExpirationService.Enabled setting to *true* in the **environmentvariables.config** file:

```
PLM4P.DBEventingExpirationService.Enabled   = true
```

The **Database Eventing Expiration Service** will delete records from the CommonEventingLog table that are a specific number of days old. To specify the number of days for expiration, set the ItemExpirationInDays entry in the CustomerSettings/Core/DatabaseEventingExpirationService node of the **CustomerSettings.config** file.

```
<add key="ItemExpirationInDays" value="14" />
<add key="SecondsToWaitBeforeStarting" value="60" />
```

Note that the Database Logger event subscriber is configurable so that it could log to a different table.

> For example, if you wanted to log the Read and Print events to one table, or PQM events to their own table, you would create a new table just like CommonEventingLog, say PQMEventingLog. Then, your subscriber factory is configured using this table name. For example:
>
> ```
> <SubscriberConfig Id="PQMAll"  EventTypes="PQM.*"  StatusIds=""
> FactoryURL="Class:Xeno.Prodika.Eventing.Subscriber.DatabaseLoggerEventSubscriberFactory
> ,GeneralServices$PQMEventingLog" … >
> ```

As such, the **Database Eventing Expiration Service** can also be configured to purge from different tables, AND to purge using different expiration days. The configuration can name multiple tables, separated by commas. Additionally, each comma separated table name can be optionally followed by the expiration number of days, separated by a colon (in the CustomerSettings/Core/DatabaseEventingExpirationService node):

ex: "CommonEventingLog,GSMEventingLog:20,PQMEventingLog:15,NPDEventingLog"

```
<add key="TableNamesAndExpirationPeriods"
value="CommonEventingLog,GSMEventingLog:20,PQMEventingLog:15,NPDEventingLog" />
```

The default or the config entry ItemExpirationInDays will be used if no specific number is given for a table. So from the above example

CommonEventingLog,GSMEventingLog:20,PQMEventingLog:15,NPDEventingLog

The expiration settings would be:

- CommonEventingLog - 20 days (using the configured value of ItemExpirationInDays)
- GSMEventingLog—20 days using the one specified as GSMEventingLog:**20 days**
- PQMEventingLog—15 days using the one specified as PQMEventingLog:**15 days**
- NPDEventingLog— 20 days - (using the configured value of ItemExpirationInDays)

Event logging information

The expiration service also logs to the system Event Log:

- When it starts the expiration run (“DatabaseEventLogExpirationService executing”)
- When it processes each table to purge (“DatabaseEventLogExpirationService executing with values tablename:{0}, expiration period in days:{1}")
- When it completes the expiration run ("DatabaseEventLogExpirationService completed");
- If it runs into an error. For example, maybe you configure it with a non-existent table (“Error occurred running DatabaseEventLogExpirationService…”)

Chapter 5—List of Raised Events

The following events are supported. Details about the information provided for each event can be found in the Event types and associated objects.xlsx document in the Extensibility Pack in the ReferenceImplementations\EventFrameworkExtensions\Documentation folder.

Application-wide Events

User Login
User Login to other application

GSM Events

Spec (or Spec Template) Create
Spec (or Spec Template) Read
Spec (or Spec Template) Edit
Spec (or Spec Template) Pre-Save
Spec (or Spec Template) Save
Spec Create From Template
Spec (or Spec Template) Revision/New Issue
Spec (or Spec Template) Copy
Spec (or Spec Template) Print
Spec (or Spec Template) Resolve Workflow
Spec (or Spec Template) Workflow
GSM Signature Document Workflow
GSM Signature Document Create
Formulation Spec Target Revision
Smart issue creation
Smart issue process completion
Testing Protocol Create
Testing Protocol Read
Testing Protocol Save
Testing Protocol Copy
Testing Protocol Edit
Global Succession event
Get Latest Revision
LIO Profile Push

PQM Events

PQM Item (or Template) Create
PQM Item (or Template) Read
PQM Item (or Template) Edit
PQM Item (or Template) Pre-Save
PQM Item (or Template) Save
PQM Item Create From Template
PQM Item (or Template) Copy
PQM Item (or Template) Workflow
PQM Item (or Template) Resolve Workflow
PQM Signature Document Workflow
PQM Signature Document Create

SCRM Events

Facility/Company (or Template) Read
Facility/Company (or Template) Create
Facility/Company CreateFromTemplate
Facility/Company (or Template) Edit
Facility/Company (or Template) Save
Facility/Company (or Template) PreSave
Sourcing Approval (or Template) Read
Sourcing Approval (or Template) Create
Sourcing Approval CreateFromTemplate
Sourcing Approval (or Template) Edit
Sourcing Approval (or Template) Save
Sourcing Approval (or Template) Workflow
Sourcing Approval (or Template) ResolveWorkflow
SCRM Signature Document Workflow
SCRM Signature Document Create
SCRM Managed Document Read
SCRM Managed Document Create
SCRM Managed Document Edit
SCRM Managed Document Save

NPD Events

NPD Activity Pre-Save
NPD Activity Save
NPD Activity Read
NPD Activity Edit
NPD Activity Create
NPD Activity Workflow
NPD Project Placed On Hold
NPD Project Released From Hold
NPD Project Pre-Save
NPD Project Save
NPD Project Read
NPD Project Edit
NPD Project Workflow
NPD Project Print
NPD Project Create