API Guide

Enterprise Events

For SharePoint 2013/2016

Version 4.13.1

January 2019

Title: *Gimmal Enterprise Events Application Programming Interface Guide*

If you have questions or comments about this publication, you can email TechnicalPublications@Gimmal.com. Be sure to identify the guide, version number, section, and page number to which you are referring. Your comments are welcomed and appreciated.

Contents

Preface

Gimmal delivers market leading content governance and compliant records solutions built on Microsoft® SharePoint® 2013 and 2016. Gimmal solutions drive user adoption and simplify information access by making information lifecycle management of content simple and transparent, ensuring consistent compliance and proactive litigation readiness enterprise-wide while lowering costs.

Gimmal products are a collection of SharePoint-native components that drive consistency in managing governance and compliance of information policies, metadata, and record routing rules to support your organization's long-term content and records management strategy. The products are easy to implement into a SharePoint environment as native features are extended from within SharePoint. User adoption is rapid, as individuals are no longer required to work outside their trusted SharePoint environment to ensure control of enterprise content.

Basic Usage

The Enterprise Events API is exposed as a Windows Communication Foundation (WCF) service. For a developer to use the API, they must consume the service in their client application using either `BasicHttpBinding` or `BasicHttpsBinding`.

The service is hosted at {*EnterpriseEventsUrl*}/WcfServices/EnterpriseEventsService.svc.

Public Member Functions

Functionality is broken into three major areas: configuration, audits, and enterprise event management.

Each basic set of functionality consists of four method types: Get All, Get Specific by ID, Save, and Remove.

Method Type	Description
Get All	Returns a simplified listing of objects.
Get Specific by ID	Returns a detailed object, obtained by ID.
Save	If the object ID is not provided or does not exists, inserts the object. Otherwise, it updates the object in the database.
Remove	Removes the object by ID.

Configuration

The configuration section of Enterprise Events sets up settings used within the product. Normally, these functions are configured at the initial install and rarely modified later.

Site Registrations

Enterprise Events is designed to manage multiple site collections, potentially across multiple farms. To allow this, Enterprise Events requires users to define site registrations.

A site registration is a site collection that Enterprise Events manages. Data, such as content types and site hierarchies, are obtained from site registrations.

GetAllSiteRegistrations

Gets all site registrations defined in Enterprise Events.

Returns

```
List<EeSiteRegistration>: A list of all site registrations defined in
Enterprise Events.  Includes encrypted username and password.
```

Example

```
// Retrieve all the Site Registrations
List<EeSiteRegistration> siteRegistrations = client.GetAllSiteRegistrations();
```

GetSiteRegistration

Gets the site registration by the specified ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the site registration.

Returns

`EeSiteRegistration`: The site registration specified by the ID.

Example

```
// Gets the site registration with the ID "0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"
EeSiteRegistration siteRegistration = client.GetSiteRegistration(new Guid("0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"));
```

Remarks

If no site registration exists with the specified ID, `null` is returned.

SaveSiteRegistration

Add or update the specified site registration.

Parameters

Name	Type	Description
siteRegistration	EeSiteRegistration	The instance of the site registration to be saved.

Returns

`EeSiteRegistration`: An updated site registration, if successful; otherwise, `null`.

Example

To add a new site registration:

```
// Create a new site registration
EeSiteRegistration newSiteRegistration = new EeSiteRegistration()
{
// Do not set the ID property when creating a new Site Registration
RegistrationType = 0,
Url = "https://mydomain.records.com/sites/legal",
SharePointSiteID = new Guid("1840A30B-4C7F-4A18-8758-BDFFEBA17D8B"),
SharePointFarmID = new Guid("D91557C5-B4AF-479D-A677-7E48A0B32068"),
UserName = @"MYDOMAIN\ujohndoe"
};

// Save the site registration using the WCF client
    newSiteRegistration = client.SaveSiteRegistration(newSiteRegistration);
```

To update an existing site registration:

```
// Retrieve the site registration to modify
EeSiteRegistration existingSiteRegistration = client.GetSiteRegistration(new Guid("2276595b-b391-42c8-97ef-e49cf68dd534"));

// Make modifications
existingSiteRegistration.Url = "https://contoso.hr.records.com/sites/legal";
// Include UserName for audits
existingSiteRegistration.UserName = @"MYDOMAIN\ujohndoe";

// Save the site registration
    existingSiteRegistration = client.SaveSiteRegistration(existingSiteRegistration);
```

Exceptions

`System.ServiceModel.FaultException<List<ValidationError>>`: if the save action fails validation (i.e. the required fields are not set).

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

RemoveSiteRegistration

Removes the site registration specified by the ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the site registration.
userName	string	The name of the user performing the deletion.

Returns

`bool`: `true` if successful; otherwise, `false`.

Example

```
    // Remove the site registration with the ID “d66e9f2e-ce6c-4803-ba54-a20ad37e03e4” by the user “MYDOMAIN\ujohndoe”
bool success = client.RemoveSiteRegistration(new Guid("d66e9f2e-ce6c-4803-ba54-a20ad37e03e4"), @"MYDOMAIN\ujohndoe");
```

Exceptions

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

Administrators

If a client wishes for users who access Enterprise Events via SharePoint to have administrative access, Enterprise Events Administrators must be configured.

An administrator is a combination of a SharePoint group and its site collection's URL/ID.

GetAllEventAdmins

Gets all administrators defined in Enterprise Events.

Returns

`List<EeEventAdmin>: A list of all administrators defined in Enterprise Events.`

Example

```
// Retrieve all administrators
List<EeEventAdmin> admins = client.GetAllEventAdmins();
```

GetEventAdmin

Gets the event administrator by the specified ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the administrator.

Returns

`EeEventAdmin`: The administrator specified by the ID.

Example

```
// Gets the event admin with the ID "0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"
EeEventAdmin admin = client.GetEventAdmin(new Guid("0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"));
```

Remarks

If no administrator exists with the specified ID, `null` is returned.

SaveEventAdmins

Add or update one or multiple administrators.

Parameters

Name	Type	Description
eventAdmins	List<EeEventAdmin>	The instances of the EeEventAdmin to be saved.

Returns

`List<EeEventAdmin>`: A list of updated administrators, if successful; otherwise, `null`.

Example

To add one or more new admins:

```
// Create an admin to be saved
```

```
EeEventAdmin admin = new EeEventAdmin()
{
// Do not set the ID property when creating new Event Admins
SharePointSiteID = new Guid("1840A30B-4C7F-4A18-8758-BDFFEBA17D8B"),
SiteCollectionUrl = "https://mydomain.records.com/sites/legal ",
ClaimsID = "RMA Administrators",
PrincipalName = "RMA Administrators"
};

    // Save the admins using the client
    List<EeEventAdmin> admins = client.SaveEventAdmins(new List<EeEventAdmin>() { admin });
```

To update an existing admin:

```
// Retrieve the admin to modify
EeEventAdmin admin = client.GetEventAdmin(new Guid("0ccf8ea9-5f57-45e3-9d3d-
e9c4f3068fe3"));
// Make modifications
admin.SiteCollectionUrl = "https://contoso.hr.records.com/sites/legal";

    // Save the admins using the client
    List<EeEventAdmin> admins = client.SaveEventAdmins(new List<EeEventAdmin>() { admin });
```

Exceptions

`System.ServiceModel.FaultException<List<ValidationError>>`: if the save action fails validation (i.e. the required fields are not set).

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

RemoveEventAdmin

Removes the administrator specified by the ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the administrator.
userName	string	The name of the user performing the deletion.

Returns

`bool`: `true` if successful; otherwise, `false`.

Example

```
    // Remove the administrator with the ID “0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3” by the user
“MYDOMAIN\ujohndoe”
    bool success = client.RemoveEventAdmin(new Guid("0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"),
@"MYDOMAIN\ujohndoe");
```

Exceptions

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

Audit Logs

The audit log section only serves a single purpose: Showing and managing audit entries. In the UI, auditing can be enabled or disabled on the Global Settings page. Not all entities are audited. Currently, the following entities, when auditing is enabled, will:

- Event Types
- Event Instances
- Event Stages
- Policy Masters
- Policy Master Association Sets

GetAllAuditLogs

Gets all audit log entries.

Returns

`List<EeAuditLog>: A list of all audit log entries that currently exist in Enterprise Events.`

Example

```
// Retrieve all audit logs
List<EeAuditLog> auditLogs = client.GetAllAuditLogs();
```

RemoveAuditLog

Removes the audit log entry specified by the ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the audit log entry.

Returns

`bool`: `true` if successful; otherwise, `false`.

Example

```
// Removes the audit log entry with the ID "0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"
bool success = client.RemoveAuditLog(new Guid("0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"));
```

Exceptions

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

RemoveAuditLogByEntityTypeAndDateRange

Removes all audit log entries of the given `entityType` with the specified `fkid` between the `startDate` and `endDate` (inclusive).

Parameters

Name	Type	Description
entityType	string	A string representing the type of audited entity. Examples: "EventTypes", "EventInstances", "EventStages", and "PolicyMasters."
fkid	string	The unique identifier of the entity that was audited.
startDate	DateTime	Start of the date range.
endDate	DateTime	End of the date range.

Returns

`bool`: `true` if successful; otherwise, `false`.

Example

// Remove all audit log entries pertaining to an Event Type with the ID "ca03e351-b0b5-4c0e-8a8e-c3b50d84f646" from January 1, 2014 to today.

```
bool success = client.RemoveAuditLogByEventTypeAndDateRange("EventTypes", "ca03e351-b0b5-4c0e-8a8e-c3b50d84f646", new DateTime(2014, 1, 1), DateTime.Now);
```

Exceptions

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

IsAuditLogEnabled

Returns whether or not audit logging is enabled for Enterprise Events.

Returns

`bool`: `true` if the audit logging is enabled; otherwise, `false`.

Example

// Determine if audit logging is enabled

```
bool isAuditLogEnabled = client.IsAuditLogEnabled();
```

SetAuditLogEnabled

Enable or disable audit logging for Enterprise Events.

Parameters

Name	Type	Description
enabled	bool	`true` to enable; `false` to disable.

Returns

`bool`: `true` if successful; otherwise, `false`.

Example

```
// Enable audit logging
bool success = client.SetAuditLogEnabled(true);
// Disable audit logging
success = client.SetAuditLogEnabled(false);
```

Exceptions

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

Manage Enterprise Events

Event Types

Event Types, also known to the end-user as types, are used to group event instances.

GetAllEventTypes

Gets all event types defined in Enterprise Events.

Returns

`List<EeEventType>: A list of all event types defined in Enterprise Events.`

Example

```
// Retrieve all event types
List<EeEventType> eventTypes = client.GetAllEventTypes();
```

Remarks

The returned event types are top-level only. They do not include associated event instances.

GetEventType

Gets the event type by the specified ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the event type.

Returns

`EeEventType`: The event type specified by the ID.

Example

```
// Gets the event type with the ID "ca03e351-b0b5-4c0e-8a8e-c3b50d84f646"
EeEventType eventType = client.GetEventType(new Guid("ca03e351-b0b5-4c0e-8a8e-c3b50d84f646"));
```

Remarks

The event type is returned with associated event instances and ACLs. If no event type exists with the specified ID, `null` is returned.

SaveEventType

Add or update the specified event type.

Parameters

Name	Type	Description
eventType	EeEventType	The instance of the event type to be saved.

Returns

`EeEventType`: An updated event type, if successful; otherwise, `null`.

Example

To add a new event type:

```
EeEventType newEventType = client.SaveEventType(new EeEventType()
{
// Do not set the ID property when creating new Event Types
Title = "Compensation Event 10001",
Description = "Compensation Event for Employee 10001",
UserName = @"MYDOMAIN\ujohndoe"
    });
```

To update an existing event type:

```
    // Retrieve the event type to modify
    EeEventType existingEventType = client.GetEventType(new Guid("4586fa06-47a6-e311-941b-
    000c29fe8c9f"));

    // Make modifications
    existingEventType.Title = "Compensation Review Event 10001A";
// Include UserName for audits
existingEventType.UserName = @"MYDOMAIN\ujohndoe";

// Save the event type
existingEventType = client.SaveEventType(existingEventType);
```

Exceptions

`System.ServiceModel.FaultException<List<ValidationError>>`: if the save action fails validation (i.e. the required fields are not set).

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

RemoveEventType

Removes the event type specified by the ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the event type.

userName	string	The name of the user performing the deletion.

Returns

bool: `true` if successful; otherwise, `false`.

Example

```
// Remove the event type with the ID “de14f339-4148-4494-be8e-63ef1684f644” by the user “MYDOMAIN\ujohndoe”
bool success = client.RemoveEventType(new Guid("de14f339-4148-4494-be8e-63ef1684f644"), @"MYDOMAIN\ujohndoe");
```

Exceptions

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

Remarks

A deletion only succeeds if the event type has no associated instances and is not associated with an event stage.

Event Type ACLs

Event Type Access Control Lists (ACLs), also known as permissions, allow or restrict access by specifying permission types on event types and their children instances.

As of date of publication, only Trigger Edit and Full Control permission types are allowed.

GetAllEventTypeAcls

Gets all event type ACLs defined in Enterprise Events.

Returns

`List<EeEventTypesAcl>: A list of all event type ACLs defined in Enterprise Events.`

Example

```
// Retrieve all event type ACLs
List<EeEventTypesAcl> eventTypeAcls = client.GetAllEventTypeAcls();
```

GetEventTypeAclsByEventTypeID

Gets all event type ACLs set for a specific event type.

Parameters

Name	Type	Description
eventTypeID	Guid	The unique identifier for the event type.

Returns

List<EeEventTypesAcl>: `A list of all event type ACLs defined for a specific event type.`

Example

```
// Retrieve all event type ACLs for the event type with ID "ca03e351-b0b5-4c0e-8a8e-c3b50d84f646"
List<EeEventTypesAcl> eventTypeAcls = client.GetEventTypeAclsByEventTypeID(new Guid("ca03e351-b0b5-4c0e-8a8e-c3b50d84f646"));
```

GetEventTypeAcl

Gets the event type ACL by the specified ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the event type ACL.

Returns

`EeEventTypesAcl`: The event type ACL specified by the ID.

Example

```
// Gets the event type ACL with the ID "ca03e351-b0b5-4c0e-8a8e-c3b50d84f646"
EeEventTypesAcl acl = client.GetEventTypeAcl(new Guid("ca03e351-b0b5-4c0e-8a8e-c3b50d84f646"));
```

Remarks

If no event type ACL exists with the specified ID, `null` is returned.

SaveEventTypeAcl

Add or update the specified event type ACL.

Parameters

Name	Type	Description
eventTypeAcl	EeEventTypesAcl	The instance of the event type ACL to be saved.

Returns

`EeEventTypesAcl`: An updated event type ACL, if successful; otherwise, `null`.

Example

To add a new event type ACL:

```
EeEventTypesAcl newEventTypeAcl = client.SaveEventTypeAcl(new EeEventTypesAcl()
{
// Do not set the ID property when creating new Event Type ACLs
EventTypeID = new Guid("4586fa06-47a6-e311-941b-000a293edc00"),
PrincipalName = "Jan Rangel",
PermissionSet = EePermissionType.TriggerEvent,
ClaimsID = "i:0#.w|devdomain\rangelj",
SiteCollectionUrl = "https://mydomain.records.com/sites/legal",
AccountType = 0,
```

```
SharePointSiteID = new Guid("4edb63e5-9a2c-4b8b-a727-138b89556304")
    });
```

To update an existing event type ACL:

```
    // Retrieve the event type ACL to modify
    EeEventTypesAcl existingAcl = client.GetEventTypeAcl(new Guid("4586fa06-47a6-e311-941b-
000c29fe8c9f"));

    // Make modifications
    existingAcl.PermissionSet = EePermissionType.FullControl;
```

// Save the event type ACL

existingAcl = client.SaveEventTypeAcl(existingAcl);

Exceptions

`System.ServiceModel.FaultException<List<ValidationError>>`: if the save action fails validation (i.e. the required fields are not set).

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

RemoveEventTypeAcl

Removes the event type ACL specified by the ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the event type ACL.
userName	string	The name of the user performing the deletion.

Returns

`bool`: `true` if successful; otherwise, `false`.

Example

```
    // Remove the event type ACL with the ID “de14f339-4148-4494-be8e-63ef1684f644” by the
user “MYDOMAIN\ujohndoe”
bool success = client.RemoveEventTypeAcl(new Guid("de14f339-4148-4494-be8e-
63ef1684f644"), @"MYDOMAIN\ujohndoe");
```

Exceptions

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

Event Instances

GetAllEventInstances

Gets all event instances defined in Enterprise Events.

Returns

`List<EeEventInstance>: A list of all event instances defined in Enterprise Events.`

Example

```
// Retrieve all event instances
List<EeEventInstance> eventInstances = client.GetAllEventInstances();
```

Remarks

The returned event instances are top-level only. They do not include associated fields or values.

GetEventInstance

Gets the event instance by the specified ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the event instance.

Returns

`EeEventInstance`: The event instance specified by the ID.

Example

```
// Gets the event instance with the ID "ca03e351-b0b5-4c0e-8a8e-c3b50d84f646"
EeEventInstance eventInstance = client.GetEventInstance(new Guid("ca03e351-b0b5-4c0e-8a8e-c3b50d84f646"));
```

Remarks

The event instance is returned with associated fields and values. If no event instance exists with the specified ID, `null` is returned.

SaveEventInstance

Add or update the specified event instance.

Parameters

Name	Type	Description
eventInstance	EeEventInstance	The instance of the event instance to be saved.

Returns

`EeEventInstance`: An updated event instance, if successful; otherwise, `null`.

Example

To add a new event instance:

```
EeEventInstance newEventInstance = new EeEventInstance()
{
// Do not set the ID property when creating new Event Instances
EventTypeID = new Guid("4586fa06-47a6-e311-941b-000c29fe8c9f"),
Title = "EmployeeComp_10001",
```

```
Description = "Compensation for Employee 10001",
Priority = 5,
TriggerDate = new DateTime(2014, 01, 01),
Fields = new List<EeEventInstanceField>()
{
new EeEventInstanceField()
{
FieldID = new Guid("3e128306-fc19-45ba-be1a-9ef0517340d3"),
FieldName = "Employee_ID",
DisplayOrder = 0,
DataType = "System.String",
Values = new List<EeEventInstanceFieldValue>()
{
new EeEventInstanceFieldValue()
{
Operator = Operator.EqualTo,
Value ="10001"
}
}
}
},
UserName = @"MYDOMAIN\ujohndoe"
    };

    // Save the instance using the client
    newEventInstance = client.SaveEventInstance(newEventInstance);
```

To update an existing event type:

```
    // Retrieve the event instance to modify
    EeEventInstance existingEventInstance = client.GetEventInstance(new Guid("4586fa06-47a6-
    e311-941b-000c29fe8c9f"));

    // Make modifications
    existingEventInstance.Title = "EmployeeComp_10001A";
// Include UserName for audits
existingEventInstance.UserName = @"MYDOMAIN\ujohndoe";

//Save the event instance
existingEventInstance = client.SaveEventInstance(existingEventInstance);
```

Exceptions

`System.ServiceModel.FaultException<List<ValidationError>>`: if the save action fails validation (i.e. the required fields are not set).

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

RemoveEventInstance

Removes the event instance specified by the ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the event instance.
userName	string	The name of the user performing the deletion.

Returns

`bool`: `true` if successful; otherwise, `false`.

Example

```
// Remove the event instance with the ID “de14f339-4148-4494-be8e-63ef1684f644” by the user “MYDOMAIN\ujohndoe”
bool success = client.RemoveEventInstance(new Guid("de14f339-4148-4494-be8e-63ef1684f644"), @"MYDOMAIN\ujohndoe");
```

Exceptions

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

Remarks

Associated EeEventInstanceFields and EeEventInstanceFieldValues are also deleted.

Event Stages

An event stage, also known to the end-user as a stage, represents a SharePoint IMP stage, and more specifically, a custom retention formula. It is a collection of event types along with a duration.

GetAllEventStages

Gets all event stages defined in Enterprise Events.

Returns

`List<EeEventStage>: A list of all event stages defined in Enterprise Events.`

Example

```
// Retrieve all event stages
List<EeEventStage> stages = client.GetAllEventStages();
```

Remarks

The returned event stages are top-level only. They do not include associated event types.

GetEventStage

Gets the event stage by the specified ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the event stage.

Returns

`EeEventStage`: The event stage specified by the ID.

Example

```
// Gets the event stage with the ID "30bee3f4-fcb6-4225-9efa-b2e2db77b2cd"
EeEventStage eventStage = client.GetEventStage(new Guid("30bee3f4-fcb6-4225-9efa-b2e2db77b2cd"));
```

Remarks

The event stage is returned with associated event types, event instances, event instance fields, and event instance field values. If no event stage exists with the specified ID, `null` is returned.

SaveEventStage

Add or update the specified event stage.

Parameters

Name	Type	Description
eventStage	EeEventStage	The instance of the event stage to be saved.

Returns

`EeEventStage`: An updated event stage, if successful; otherwise, `null`.

Example

To add a new event stage:

```
EeEventStage newEventStage = client.SaveEventStage(new EeEventStage()
{
// Do not set the ID property when creating new event stages
Title = "Compensation Review Stage 10001",
Description = "Compensation Review stage",
EventTypes = new List<EeEventType>()
{
new EeEventType()
{
ID = new Guid("dacc3d47-47a6-e311-941b-000c29fe8c9f")
}
},
DateCalcType = EeEventStageDateCalculationType.MinAll,
```

```
Duration = 7,
DurationType = EeDurationType.Years,
UserName = @"MYDOMAIN\ujohndoe"
});
```

To update an existing event stage:

```
    // Retrieve the event stage to modify
    EeEventStage existingEventStage = client.GetEventStage(new Guid("80c3111b-d5a7-e311-941b-000c29fe8c9f"));

    // Make modifications
    existingEventStage.Title = "Compensation Review Stage 10001A";
// Include UserName for audits
existingEventStage.UserName = @"MYDOMAIN\ujohndoe";

// Save the event stage
    existingEventStage = client.SaveEventStage(existingEventStage);
```

Exceptions

`System.ServiceModel.FaultException<List<ValidationError>>`: if the save action fails validation (i.e. the required fields are not set).

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

RemoveEventStage

Removes the event stage specified by the ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the event stage.
userName	string	The name of the user performing the deletion.

Returns

`bool`: `true` if successful; otherwise, `false`.

Example

```
    // Remove the event stage with the ID “53606ee2-b2c2-4b36-acd6-2ab940ae0fa7” by the user “MYDOMAIN\ujohndoe”
bool success = client.RemoveEventStage(new Guid("53606ee2-b2c2-4b36-acd6-2ab940ae0fa7"), @"MYDOMAIN\ujohndoe");
```

Exceptions

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

Remarks

A deletion only succeeds if the event stage is not associated with a policy master and is not in use by File Plan Builder (see Policy Stage Usage).

Policy Stage Usage

Policy stage usage is specifically used by File Plan Builder. At this point, no other external source should use it.

A policy stage usage is a way for an external application to inform Enterprise Events that it is using an event stage for its purposes. If a policy stage usage exists for a specific event stage, the event stage cannot be deleted.

In File Plan Builder, when a disposition phase is set to use Enterprise Events, and an event stage is selected, File Plan Builder will create a policy stage usage for the selected event stage. When the event stage is no longer used in any disposition phase, File Plan Builder will delete the policy stage usage.

Policy Masters

A policy master, also known to the end-user as a policy, represents a SharePoint retention schedule. It consists of a list of event stages, each associated with a disposition action.

GetAllPolicyMasters

Gets all policy masters defined in Enterprise Events.

Returns

`List<EePolicyMaster>: A list of all policy masters defined in Enterprise Events.`

Example

```
// Retrieve all policy masters
List<EePolcyMaster> policies = client.GetAllPolicyMasters();
```

Remarks

The returned policy masters are top-level only. They do not include associated event stages nor actions.

GetPolicyMaster

Gets the policy master by the specified ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the policy master.

Returns

`EePolicyMaster`: The policy master specified by the ID.

Example

```
// Gets the policy master with the ID "0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"
```

```
EePolicyMaster policy = client.GetPolicyMaster(new Guid("0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"));
```

Remarks

The policy master is returned with associated event stages and actions. If no policy master exists with the specified ID, `null` is returned.

SavePolicyMaster

Add or update the specified policy master.

Parameters

Name	Type	Description
policyMaster	EePolicyMaster	The instance of the policy master to be saved.

Returns

`EePolicyMaster`: An updated policy master, if successful; otherwise, `null`.

Example

To add a new policy master:

```
// Create the new Policy Master
EePolicyMaster policyMaster = new EePolicyMaster()
{
// Do not set the ID property when creating new Policy Masters
Title = "Compensation Policy Master 10001",
Description = "",
Stages = new List<EePolicyMasterStage>()
{
        new EePolicyMasterStage()
        {
                PolicyStageID = new Guid("30bee3f4-fcb6-4225-9efa-b2e2db77b2cd"),
                StageNumber = 0,
                Action =
"Microsoft.Office.RecordsManagement.PolicyFeatures.Expiration.Action.DeletePreviousDrafts",
                ActionXml = "<action type=\"action\"
id=\"Microsoft.Office.RecordsManagement.PolicyFeatures.Expiration.Action.DeletePreviousDrafts
\" />"
        },
        new EePolicyMasterStage()
        {
                PolicyStageID = new Guid("f24708a4-01d6-e311-9485-000c29ceeb83"),
                StageNumber = 1,
```

```
                Action = "StartWorkflow",
                ActionXml = "<action type="workflow" />",
                AdditionalData = "My Approval Workflow Name"
            }

},
UserName = @"MYDOMAIN\ujohndoe"
};

// Save the PolicyMaster
    policyMaster = client.SavePolicyMaster(policyMaster);
```

When modifying an existing Policy Master in the system:

```
    // Retrieve the policy master to modify
    EePolicyMaster existingPolicyMaster = client.GetPolicyMaster(new Guid("4586fa06-47a6-e311-941b-000c29fe8c9f"));

    // Make modifications
    existingPolicyMaster.Title = "Compensation Policy Master 10001A";
existingPolicyMaster.UserName = @"MYDOMAIN\ujohndoe";

// Save the policy master
    existingPolicyMaster = client.SavePolicyMaster(existingPolicyMaster);
```

Exceptions

`System.ServiceModel.FaultException<List<ValidationError>>`: if the save action fails validation (i.e. the required fields are not set).

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

RemovePolicyMaster

Removes the policy master specified by the ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the policy master.
userName	string	The name of the user performing the deletion.

Returns

`bool`: `true` if successful; otherwise, `false`.

Example

```
    // Remove the policy master with the ID “9bf73ee2-96e5-483a-bc9b-f0bc9584c638” by the user “MYDOMAIN\ujohndoe”
```

```
bool success = client.RemovePolicyMaster(new Guid("9bf73ee2-96e5-483a-bc9b-f0bc9584c638"), @"MYDOMAIN\ujohndoe");
```

Exceptions

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

Remarks

A deletion only succeeds if the policy master is not associated with a policy association set.

Policy Master Association Sets

A policy master association set, also known to the end-user as an association, represents a record and/or non-record policy applied to a list of targets (policy master associations).

GetAllPolicyMasterAssociationSets

Gets all policy master association sets defined in Enterprise Events.

Returns

`List<EePolicyMasterAssociationSet>`: A list of all policy master association sets defined in Enterprise Events.

Example

```
// Retrieve all policy master association sets
List<EePolicyMasterAssociationSet> associationSets = client.GetAllPolicyMasterAssociationSets();
```

Remarks

The returned policy master association sets are top-level only. They do not include associated policy master associations.

GetPolicyMasterAssociationSet

Gets the policy master association set by the specified ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the policy master association set.

Returns

`EePolicyMasterAssociationSet`: The policy master association set specified by the ID.

Example

```
// Gets the policy master association set with the ID "0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"
EePolicyMasterAssociationSet associationSet = client.GetPolicyMasterAssociationSet(new Guid("0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"));
```

Remarks

The policy master association set is returned with associated policy master associations, policy masters (both record and non-record), and policy stages. If no policy master association set exists with the specified ID, `null` is returned.

SavePolicyMasterAssociationSet

Add or update the specified policy master association set.

Parameters

Name	Type	Description
associationSet	EePolicyMasterAssociationSet	The instance of the policy master association set to be saved.

Returns

`EePolicyMasterAssociationSet`: An updated policy master association set, if successful; otherwise, `null`.

Example

To add a new policy master association set:

```
// Create the new policy master association set
EePolicyMasterAssociationSet associationSet = new EePolicyMasterAssociationSet()
{
// Do not set the ID property when creating new Policy Master Association Sets
Title = "Compensation Policy Association Set 10001",
Description = "",
NonRecordPolicyMasterID = new Guid("0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"),
RecordPolicyMasterID = null,
RecordSameAsNonRecord = true,
PendingDissocation = false,
Associations = new List<EePolicyMasterAssociation>()
{
        new EePolicyMasterAssociation()
        {
                SiteUrl = "https://mydomain.records.com/sites/legal",
                SiteID = new Guid("035D22C8-8607-4108-94C1-D6CCEDB6DD7B"),
                WebID = new Guid("FEB14491-3216-49C3-94C8-5B6F36D69048"),
ContentTypeID = null,
                LocationID = new Guid("E83438D8-E691-4A99-AF7F-6912B6386A55"),
PendingDissociation = false
        },
        new EePolicyMasterAssociation()
```

```
        {
                SiteUrl = "https://mydomain.records.com/sites/legal",
                SiteID = new Guid("035D22C8-8607-4108-94C1-D6CCEDB6DD7B"),
                WebID = new Guid("FEB14491-3216-49C3-94C8-5B6F36D69048"),
ContentTypeID = "0x010100E1F0ECEADA9D7944AB4B2A38B6B0D85E",
                LocationID = null,
PendingDissociation = false
        }
},
UserName = @"MYDOMAIN\ujohndoe"
};

// Save the association set
    associationSet = client.SavePolicyMasterAssociationSet(associationSet);
```

When modifying an existing Policy Master in the system:

```
    // Retrieve the policy master association set to modify
    EePolicyMasterAssociationSet existingAssociationSet =
client.GetPolicyMasterAssociationSet(new Guid("4586fa06-47a6-e311-941b-000c29fe8c9f"));

    // Make modifications
    existingAssociationSet.RecordPolicyMasterID = new Guid("212C8F2C-C5D7-E311-9487-
    000C29CEEB83");
    existingAssociationSet.RecordSameAsNonRecord = false;
existingAssociationSet.UserName = @"MYDOMAIN\ujohndoe";

// Save the policy master association set
    existingAssociationSet = client.SavePolicyMasterAssociationSet(existingAssociationSet);
```

Exceptions

`System.ServiceModel.FaultException<List<ValidationError>>`: if the save action fails validation (i.e., the required fields are not set).

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

DissociatePolicyMasterAssociationSet

Marks the policy master association set for dissociation, then deletion, by the specified ID.

Policy master associations sets are never directly deleted. Enterprise Event's Association timer job removes the retention schedule from all targets of the association set. Upon successful removal, the timer job deletes the policy master associations and policy master association set.

Parameters

Name	Type	Description

id	Guid	The unique identifier for the policy master association set.

Returns

`EePolicyMasterAssociationSet`: An updated policy master association set, if exists and successful; otherwise, `null`.

Example

```
// Marks the policy master association set with the ID "0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3" for dissociation
EePolicyMasterAssociationSet associationSet = client.DissociatePolicyMasterAssociationSet(new Guid("0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"), @"MYDOMAIN\ujohndoe");
```

Exceptions

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

Remarks

If no policy master association set exists with the specified ID, or an error occurs during save, `null` is returned.

RemovePolicyMasterAssociationSet

Removing a policy master association set is specifically used by Enterprise Event's Association timer job. At this point, no other external source should use it.

Policy Master Associations

A policy master association, also known to the end-user as a target, represents a location or content type to which a retention schedule (policy master) is created.

GetPolicyMasterAssociation

Gets the policy master association by the specified ID.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the policy master association.

Returns

`EePolicyMasterAssociation`: The policy master association specified by the ID.

Example

```
// Gets the policy master association with the ID "0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"
EePolicyMasterAssociation association = client.GetPolicyMasterAssociation(new Guid("0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"));
```

Remarks

If no policy master association exists with the specified ID, `null` is returned.

DissociatePolicyMasterAssociation

Marks the policy master association for dissociation, then deletion, by the specified ID.

Policy master associations are never directly deleted. Enterprise Event's Association timer job removes the retention schedule from the target. Upon successful removal, the timer job deletes the policy master association.

Parameters

Name	Type	Description
id	Guid	The unique identifier for the policy master association.

Returns

`EePolicyMasterAssociation`: An updated policy master association, if exists and successful; otherwise, `null`.

Example

```
// Marks the policy master association with the ID "0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3" for dissociation
EePolicyMasterAssociation association = client.DissociatePolicyMasterAssociation(new Guid("0ccf8ea9-5f57-45e3-9d3d-e9c4f3068fe3"), @"MYDOMAIN\ujohndoe");
```

Exceptions

`System.ServiceModel.FaultException<DbSaveError>`: if committing to the database fails (i.e. foreign key constraint errors).

Remarks

If no policy master association exists with the specified ID, or an error occurs during save, `null` is returned.

RemovePolicyMasterAssociation

Removing a policy master association is specifically used by Enterprise Event's Association timer job. At this point, no other external source should use it.

Event Management to Enterprise Events Migration

Clients may wish to migrate their current Compliance Suite's Event Management data to Enterprise Events. Using the API, this task is simple.

Table Mappings

The following tables outline the database table mappings between Compliance Suite's Event Management and Enterprise Events.

Gimmal_Event.dbo.EVENTS		Gimmal_EE.dbo.EventTypes	Comments
EventID		ID	int to uniqueidentifier
Name		Name	
Description		Description	
AttributeFieldName			No longer needed
UseAcl		UseUserDefinedACLs	

Gimmal_Event.dbo. EVENT_INSTANCES		Gimmal_EE.dbo. EventInstances	Comments
EventInstanceID		ID	int to uniqueidentifier
EventID		EventTypeID	int to uniqueidentifier
AttributeFieldValue			No longer needed
Date		TriggerDate	
		Name	
		Description	
		Reason	
		Priority	
		CreateDate	Auto-gen by database

Gimmal_Event.dbo. EVENTS_ACL		Gimmal_EE.dbo. EventTypesACL	Comments
		ID	
EventID		EventTypeID	int to uniqueidentifier
PrincipalName		PrincipalName	Requires translation
PermissionSet		PermissionSet	Requires translation
ClaimsId		ClaimsID	

		SiteCollectionUrl	
		AccountType	0 for user, 1 for group
		SharePointSiteID	

Migration Comments

Migrating EVENTS to EventTypes is a simple column-to-column data migration.

Migrating EVENT_INSTANCES to EventInstances requires a Name and Priority. The old columns map nicely to new columns.

Migrating EVENTS_ACL to EventTypesACL requires a bit more work.

- The PrincipalName in Compliance Suite's Event Management is normally a claims string (e.g. – "i:0#.w|mydomain\ujohndoe") or NTLM login name (e.g. – "MYDOMAIN\ujohndoe"). The PrincipalName in Enterprise Events is normally the user's actual name (e.g. – "John Doe"). It is up to the client if obtaining the user's actual name is important. For Enterprise Events, we only use the actual najme to display it nicely in the people pickers.
- The PermissionSet in Compliance Suite's Event Management has more options than the current version of Enterprise Events. Event Management ACLs can be ReadEvent (1), UpdateEvent (2), DeleteEvent (4), TriggerEvent (8), or ManagePermissions (16). While Enterprise Events' enum contains the same as Event Management (and a couple more), only two are valid for this version: TriggerEvent (8) and FullControl (32).

  The following table indicates the desired translation from Event Management Permission Sets to Enterprise Events' Permission Sets:

Event Management		Enterprise Events
ReadEvent		Ignored
UpdateEvent		TriggerEvent
DeleteEvent		FullControl
TriggerEvent		TriggerEvent
ManagePermissions		FullControl

Example

First, add an assembly reference for GimmalSoft.Common.SharePoint, GimmalSoft.ComplianceSuite.Common.SharePoint, GimmalSoft.ComplianceSuite.EventManagement.SharePoint, and Microsoft.SharePoint. For a WFE, these should all be in the GAC.

Second, add a service reference to the EnterpriseEventsService.svc.

```
using EventsMigration.EeService;
using GimmalSoft.Common;
using GimmalSoft.ComplianceSuite.Common.BusinessEntity;
using GimmalSoft.ComplianceSuite.EventManagement.ServiceApplication;
using GimmalSoft.ComplianceSuite.EventManagement.ServiceApplication.Proxy;
using Microsoft.SharePoint.Administration;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.ServiceModel;

namespace Gimmal.EnterpriseEvents.Migration
{
    public class EventsMigration
    {
        private GimmalEventManagementServiceClient _eventManagementClient;
        private EnterpriseEventsServiceClient _eeWcfClient;

        public void Migrate()
        {
```

```
        var eventManagementService = SPFarm.Local.Services.GetValue<GimmalEventManagementService>();
        if (eventManagementService == null)
        {
            throw new NullReferenceException("Gimmal Compliance Suite - Event Management is not installed.");
        }

        if (SPDatabaseHelper.Exists<GimmalEventManagementServiceDatabase>())
        {
            // This will throw an exception if none are found.  We want that.
            var svcapp = eventManagementService.Applications.OfType<GimmalEventManagementServiceApplication>().First();
            var proxy = GimmalEventManagementServiceApplication.GetProxyForApplicationName(svcapp.Name);
            _eventManagementClient = new GimmalEventManagementServiceClient(SPHelper.GetDefaultServiceContext(), proxy);
            _eeWcfClient = new EnterpriseEventsServiceClient("BasicHttpBinding_IEnterpriseEventsService");

            CreateEventTypes();
        }
        else
        {
            Console.WriteLine("WARNING: GimmalEventManagementServiceDatabase not found.");
        }

        Console.WriteLine("Press any key to continue...");
        Console.ReadKey();
    }

    private void CreateEventTypes()
```

```
{
    // Get all event types from Event Management
    var gsEventTypes = _eventManagementClient.ListEvents(null);
    foreach (var gsEventType in gsEventTypes)
    {
        // Map the old event type to the new event type
        EeEventType eeEventType = new EeEventType()
        {
            Title = gsEventType.Name,
            Description = gsEventType.Description,
            UseUserAcls = gsEventType.UseACL
        };

        // Save the event type to Enterprise Events and get an updated version (includes ID)
        eeEventType = TryAction(() => _eeWcfClient.SaveEventType(eeEventType));

        CreateEventInstances(gsEventType, eeEventType);
        CreateEventTypeAcls(gsEventType, eeEventType);
    }
}

private void CreateEventInstances(EVENT gsEventType, EeEventType eeEventType)
{
    // Get all event instances from Event Management.
    // Pass in an admin's user name; otherwise, not all results will be obtained.
    var gsEventInstances = _eventManagementClient.ListEventInstances(gsEventType.EventID, @"DEVDOMAIN\SharePoint");
```

```
        foreach (var gsEventInstance in gsEventInstances)
        {
            // Map the old event instance to the new event instance
            EeEventInstance eeEventInstance = new EeEventInstance()
            {
                Title = gsEventInstance.EventInstanceID.ToString(CultureInfo.InvariantCulture),
                EventTypeID = eeEventType.ID,
                TriggerDate = gsEventInstance.Date,
                Priority = 5,
                Fields = new List<EeEventInstanceField>()
            };

            // Save the event instance to Enterprise Events
            TryAction(() => _eeWcfClient.SaveEventInstance(eeEventInstance));
        }
    }

    private void CreateEventTypeAcls(EVENT gsEventType, EeEventType eeEventType)
    {
        // Get all site registrations from Enterprise Events
        var eeSiteRegs = _eeWcfClient.GetAllSiteRegistrations();
        if (eeSiteRegs.Count > 0)
        {
            // Get all ACLs from Event Management.  Pass in an admin's user name; otherwise, not all results will be obtained.
            var gsEventAcls = _eventManagementClient.ReadEventSecurity(gsEventType.EventID, @"DEVDOMAIN\SharePoint");
            foreach (var gsAcl in gsEventAcls.Where(gsAcl => gsAcl.PermissionSet > 1))
```

```
        {
            // Few things of note:
            // * Account Type of 0 is a user; 1 is a group.
            // * The principal name should be converted to a friendly display name (e.g. - "Jan Rangel")
            // * The SharePointSiteID and SiteCollectionUrl need to be obtained/specified.  Here, we
            //   assume we'll use the first site registration.
            EeEventTypesAcl eeAcl = new EeEventTypesAcl()
            {
                AccountType = 0,
                EventTypeID = eeEventType.ID,
                ClaimsID = gsAcl.ClaimsId,
                PrincipalName = gsAcl.PrincipalName,
                PermissionSet = ConvertPermissionType(gsAcl.PermissionSet),
                SharePointSiteID = eeSiteRegs[0].SharePointSiteID,
                SiteCollectionUrl = eeSiteRegs[0].Url
            };

            // Save the ACL to Enterprise Events
            TryAction(() => _eeWcfClient.SaveEventTypeAcl(eeAcl));
        }
    }
    else
    {
        Console.WriteLine("ERROR: No site registrations found.");
    }
}
```

```
    private static EePermissionType ConvertPermissionType(int eventManagementPermissionLevel)
    {
        switch (eventManagementPermissionLevel)
        {
            case 1:  // Read
                throw new ArgumentOutOfRangeException("eventManagementPermissionLevel", "Read is an invalid permission level for
Enterprise Events.");
            case 2:  // Update
            case 8:  // Trigger
                return EePermissionType.TriggerEvent;
            case 4:  // Delete
            case 16: // Manage Perm
                return EePermissionType.FullControl;
            default:
                throw new ArgumentOutOfRangeException("eventManagementPermissionLevel", "Invalid permission level.");
        }
    }

    private static T TryAction<T>(Func<T> action)
    {
        try
        {
            return action();
        }
        catch (FaultException<DbSaveError> dbex)
```

```
            {
                Console.WriteLine("Save error: ");
                foreach (var exMsg in dbex.Detail.ExceptionMessages)
                {
                    Console.WriteLine("  " + exMsg);
                }
            }
            catch (FaultException<List<ValidationError>> fex)
            {
                Console.WriteLine("Save error: ");
                foreach (var ve in fex.Detail)
                {
                    Console.WriteLine("  " + ve.ErrorMessage);
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Unhandled exception: " + ex.Message);
            }

            return default(T);
        }
    }
}
```

Coding Samples

This section contains code for common scenarios.

SharePoint Web App Sites to Site Registrations

Assume you want to create a site registration for all site collections that exist in a web application. Also, assume you want to create an Enterprise Events Administrator for all Compliance Suite groups within any of those site collections.

```
using EventsMigration.EeService;
using GimmalSoft.Common;
using Microsoft.SharePoint;
using Microsoft.SharePoint.Administration;
using Microsoft.SharePoint.Administration.Claims;
using System;
using System.Collections.Generic;
using System.Linq;
using System.ServiceModel;

namespace Gimmal.EnterpriseEvents.Samples
{
   public class WebAppToSiteReg
   {
      private const string EeSvcPath = "WcfServices/EnterpriseEventsService.svc";
      private const string UserName = "sharepoint@devdomain.gimmal.com";
      private const string Password = "asdf";

      private readonly string[] _adminGroups = { "RMA Administrators", "RMA Local Records Officers", "RMA Records Managers" };
```

```
        /// <summary>
        /// Creates site registrations for all site collections in the specified SharePoint web application.
        /// </summary>
        /// <param name="webAppUrl">
        /// The SharePoint web application URL.
        /// <example>"https://adfsdev.devdomain.gimmal.com:22222"</example>
        /// </param>
        /// <param name="eeUrl">
        /// The Enterprise Events web applilcation URL.
        /// <example>"http://dev.devdomain.gimmal.com"</example>
        /// </param>
        /// <param name="eeHttpPort">
        /// The Enterprise Events web application HTTP port.
        /// <example>5081</example>
        /// </param>
        /// <param name="eeHttpsPort">
        /// The Enterprise Events web application HTTP port.
        /// <example>5082</example>
        /// </param>
        public void Migrate(string webAppUrl, string eeUrl, int? eeHttpPort, int? eeHttpsPort)
        {
            SPWebApplication webApp = SPWebApplication.Lookup(new Uri(webAppUrl));
            var client = WcfClientHelper.GetServiceClient<EnterpriseEventsServiceClient, IEnterpriseEventsService>(eeUrl, EeSvcPath,
eeHttpPort, eeHttpsPort);
```

```
// Attempt to the ADFS URL from the Web App.  This URL is specified via PowerShell.
string adfsProviderUrl = GetAdfsUrl(webApp);

// If adfsProviderUrl is null, assume the site is NTLM.  A different checking mechanism could be used.
// EnterpriseEvents.Web has a registration type enum.  The DTO doesn't.  The following are the values:
// SharePoint2013KerbNtlm = 0, SharePoint2013Adfs = 1, SharePoint2010KerbNtlm = 2, SharePoint2010Adfs = 3
int registrationType = string.IsNullOrWhiteSpace(adfsProviderUrl) ? 0 : 1;

// Iterate each site collection
foreach (SPSite spSite in webApp.Sites)
{
    // Map the site collection to a site registration
    var eeSiteRegistrations = new EeSiteRegistration()
    {
        Url = spSite.Url,
        UserName = UserName,
        Password = Password,
        SharePointSiteID = spSite.ID,
        RegistrationType = (short)registrationType,
        AdfsUrl = adfsProviderUrl,
        SharePointFarmID = webApp.Farm.Id
    };
    // Save the site registration
    TryAction(() => client.SaveSiteRegistration(eeSiteRegistrations));

    // Add admins if any of the three Compliance Suite groups exist in the site collection
```

```
            var eeAdmins = new List<EeEventAdmin>();
            foreach (SPGroup group in spSite.RootWeb.Groups.OfType<SPGroup>().Where(o => _adminGroups.Contains(o.Name)))
            {
                eeAdmins.Add(new EeEventAdmin()
                {
                    PrincipalName = group.Name,
                    ClaimsID = group.Name,
                    SharePointSiteID = spSite.ID,
                    SiteCollectionUrl = spSite.Url
                });
            }
            // Save the admins if were found
            if (eeAdmins.Count > 0)
            {
                TryAction(() => client.SaveEventAdmins(eeAdmins));
            }
        }

        Console.Read();
    }

    public static string GetAdfsUrl(SPWebApplication webApp)
    {
        if (webApp.IisSettings.Count > 0)
        {
            var iisSettings = webApp.IisSettings.Count > 1 ? webApp.IisSettings[SPUrlZone.Default] : webApp.IisSettings[0];
```

```
            if (iisSettings.ClaimsAuthenticationProviders != null)
            {
                SPTrustedAuthenticationProvider sptap =
iisSettings.ClaimsAuthenticationProviders.OfType<SPTrustedAuthenticationProvider>().FirstOrDefault();
                if (sptap != null)
                {
                    var provider = SPSecurityTokenServiceManager.Local.TrustedLoginProviders.GetProviderByName(sptap.DisplayName);
                    return provider.ProviderUri.ToStringSafe().Replace("/adfs/ls", string.Empty).EnsureEndsWith("/");
                }
            }
        }
        return null;
    }

    private static T TryAction<T>(Func<T> action)
    {
        try
        {
            return action();
        }
        catch (FaultException<DbSaveError> dbex)
        {
            Console.WriteLine("Save error: ");
            foreach (var exMsg in dbex.Detail.ExceptionMessages)
            {
                Console.WriteLine("  " + exMsg);
```

```
            }
        }
        catch (FaultException<List<ValidationError>> fex)
        {
            Console.WriteLine("Save error: ");
            foreach (var ve in fex.Detail)
            {
                Console.WriteLine("  " + ve.ErrorMessage);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Unhandled exception: " + ex.Message);
        }

        return default(T);
    }
  }
}
```

Enterprise Events Service WCF Interface

```
public interface IEnterpriseEventsService
{
    List<EeEventType> GetAllEventTypes();
    EeEventType GetEventType(Guid id);
    EeEventType SaveEventType(EeEventType eventType);
    bool RemoveEventType(Guid id, string userName);

    List<EeEventTypesAcl> GetAllEventTypeAcls();
    List<EeEventTypesAcl> GetEventTypeAclsByEventTypeID(Guid eventTypeID);
    EeEventTypesAcl GetEventTypeAcl(Guid id);
    EeEventTypesAcl SaveEventTypeAcl(EeEventTypesAcl eventTypeAcl);
    bool RemoveEventTypeAcl(Guid id, string userName);

    List<EeEventInstance> GetAllEventInstances();
    EeEventInstance GetEventInstance(Guid id);
    EeEventInstance SaveEventInstance(EeEventInstance eventInstance);
    bool RemoveEventInstance(Guid id, string userName);

    List<EeEventStage> GetAllEventStages();
    EeEventStage GetEventStage(Guid id);
    EeEventStage SaveEventStage(EeEventStage eventStage);
    bool RemoveEventStage(Guid id, string userName);

    bool SavePolicyStageUsage(EePolicyStageUsage policyStageUsage);
    bool RemovePolicyStageUsage(EePolicyStageUsage policyStageUsage);

    List<EePolicyMaster> GetAllPolicyMasters();
    EePolicyMaster GetPolicyMaster(Guid id);
    EePolicyMaster SavePolicyMaster(EePolicyMaster policyMaster);
    bool RemovePolicyMaster(Guid id, string userName);

    List<EePolicyMasterAssociationSet> GetAllPolicyMasterAssociationSets();
    EePolicyMasterAssociationSet GetPolicyMasterAssociationSet(Guid id);
```

```
    EePolicyMasterAssociationSet SavePolicyMasterAssociationSet(EePolicyMasterAssociationSet
assocationSet);
    EePolicyMasterAssociationSet DissociatePolicyMasterAssocationSet(Guid id, string userName);
    bool RemovePolicyMasterAssocationSet(Guid id, string userName);

    EePolicyMasterAssociation GetPolicyMasterAssociation(Guid id);
    EePolicyMasterAssociation DissociatePolicyMasterAssociation(Guid id, string userName);
    bool RemovePolicyMasterAssociation(Guid id, string userName);

    List<EeEventAdmin> GetAllEventAdmins();
    EeEventAdmin GetEventAdmin(Guid id);
    List<EeEventAdmin> SaveEventAdmins(List<EeEventAdmin> eventAdmins);
    bool RemoveEventAdmin(Guid id, string userName);

    List<EeSiteRegistration> GetAllSiteRegistrations();
    EeSiteRegistration GetSiteRegistration(Guid id);
    EeSiteRegistration SaveSiteRegistration(EeSiteRegistration siteRegistration);
    bool RemoveSiteRegistration(Guid id, string userName);

    bool IsAuditLogEnabled();
    bool SetAuditLogEnabled(bool enabled);
    List<EeAuditLog> GetAllAuditLogs();
    bool RemoveAuditLog(Guid id);
    bool RemoveAuditLogByEntityTypeAndDateRange(string entityType, string fkid, DateTime
startDate, DateTime endDate);
}
```

Class Reference

The following classes are used by the service. Most classes exist in the GimmalSoft.EnterpriseEvents.Common assembly and in the GimmalSoft.EnterpriseEvents.Common namespace; most should also come across when consuming the WCF service.

Classes

System.Data.Entity.Validation.DbEntityValidationResult

Assembly: EntityFramework.dll

This class is used in conjunction with DbSaveError.

Properties

Type	Name	Description
List<DbValidationError>	ValidationErrors	Gets validation errors. Never null.

GimmalSoft.EnterpriseEvents.Web.Models.DbSaveError

Assembly: GimmalSoft.EnterpriseEvents.Web.dll

This class is used when a database commit error occurs. Normally happens for foreign key constraint violations.

Properties

Type	Name	Description
List<string>	ExceptionMessages	
List<string>	ExceptionStackTraces	
List<DbEntityValidationResult>	ValidationResults	

System.Data.Entity.Validation.DbValidationError

Assembly: EntityFramework.dll

This class is used in conjunction with DbSaveError.

Properties

Type	Name	Description
string	ErrorMessage	Gets the validation error message.
string	PropertyName	Gets the name of the invalid property.

EeAuditLog

Represents an audit log entity.

Properties

Type	Name	Description	Example Values
Guid	ID	Gets the unique ID for the audit log entry.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
string	FKID	Gets the foreign key ID of the entity that was audited. For example, if the audit entry was for an event stage, the value will be the string representation of the event stage's ID.	"7e056bfa-4e8e-493c-b860-a48ab3abc13a"
short	AuditType	Gets the type of the audit. Maps to System.Data.Entity.EntityState.	We only use: 4 = Added 8 = Deleted 16 = Modified
string	UserName	Gets the name of the user. This will be the SharePoint User when running within SharePoint and the local Admin when running locally.	"ujohndoe"
string	SPUserName	Gets the name of the SharePoint user. This only applies when running in SharePoint.	"SPDOMAIN\\ujohndoe"
string	EntityType	Gets the type of entity audited.	Example values are `"EntityTypes"` for Event Type or `"PolicyMasters"` for Policy Master.
string	BeforeData	Gets information describing the Entity's data before the audit.	The entity serialized as XML.

string	AfterData	Gets information describing the Entity's data after the audit.	The entity serialized as XML.
DateTime	AuditDate	Gets the audit date.	

EeEventAdmin

Represents an Enterprise Events Administrator.

Properties

Type	Name	Description	Example Values
Guid	ID	Gets or sets the unique ID for the administrator. When creating a new instance, do not set; this value will be auto-generated by the database.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
string	PrincipalName	Gets or sets the principal name of the admin. Should be a friendly display name.	"SharePoint Report Readers"
string	ClaimsID	Gets or sets the claims identity string. Admins are only SharePoint groups. Therefore, it should be the group name.	"RMA Administrators"
string	SiteCollectionUrl	Gets or sets URL of the site collection for which the admin is a member.	"https://engineering.contoso.com/sites/legal"
Guid	SharePointSiteID	Gets or sets ID of the site collection for which the admin is a member.	new Guid("2c86a4f2-d9de-41c7-a358-49f750bb6233")

EeEventInstance

Represents an instance.

Properties

Type	Name	Description	Example Values
Guid	ID	Gets or sets the unique ID for the instance. When creating a new instance, do not set; this value will be auto-generated by the database.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
Guid	EventTypeID	Gets or sets the ID of the parent event type.	new Guid("523bda49-5419-4b45-b5c6-2742daa0f4fe")
string	EventTypeTitle	Gets the event type title. Setting this has no effect on saving. It is purely for information purposes.	"Anniversary Event Type"
string	Title	Gets or sets the title of the instance.	"Anniversary Event Instance"
string	Description	Gets or sets the description of the instance.	"An event instance representing the Anniversary of an event."
DateTime?	TriggerDate	Gets or sets the trigger date for the instance.	new DateTime(2014,3,11) // March 11, 2014
string	Reason	Gets or sets the reason for the trigger date.	"The date that triggers when affected records will be removed."
List<EeEventInstanceField>	Fields	Gets or sets the event instance fields associated with this instance.	
byte	Priority	Gets or sets the priority of the event instance.	1 // Highest 5 // Medium/Default 9 // Lowest
DateTime	CreateDate	Gets the create date.	new DateTime(2014,3,11) // March 11, 2014
string	UserName	Get or sets the UserName. This is used for logging purposes.	@"MYDOMAIN\ujohndoe"

EeEventInstanceField

Represents the attributes of an instance.

Methods

Return Type	Name	Parameters	Description	Example usage
void	ParseFieldName	out string displayName out string internalName	Gets the display name and internal name from the field name.	string displayName; string internalName; savedEventInstance.Fields[0]. ParseFieldName(out displayName, out internalName);

Properties

Type	Name	Description	Example Values
Guid	ID	Gets or sets the unique ID for the field. When creating a new instance, do not set; this value will be auto-generated by the database.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
Guid	FieldID	Gets or sets the SPFieldID. This is a Field ID of a site column obtained from the SharePoint site configured in Global Settings.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
string	FieldName	Gets or sets the field name.	"Employee ID"
int	DisplayOrder	Gets or sets the UI display order of the field. This is in respect to the other fields in the instance.	0 thru 2
string	DataType	Gets or sets the type of the data.	"System.String" and "System.DateTime"

List<EeEventInstanceFieldValue>	Values	Gets or sets the field values associated with this field.	

EeEventInstanceFieldValue

Represents the value of an attribute of an instance.

Public Attributes

Type	Name	Description	Value
const string	ValueDelimiter	The value delimiter.	"#;#"

Properties

Type	Name	Description	Example Values
Guid	ID	Gets or sets the unique ID for the field value. When creating a new instance, do not set; this value will be auto-generated by the database.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
Operator	Operator	Gets or sets the operator.	Operator.EqualTo Operator.Contains
string	Value	Gets or sets the value.	The value(s), possibly separated by the ValueDelimiter. This is dependent on the value of Operator. For instance, if Operator is `Operator.Equals`, enter `"Jack"`. For instance, if Operator is Operator.IsBetweenInclusive, enter "1.5#;#7.83".

EeEventStage

Represents a stage.

Properties

Type	Name	Description	Example Values
Guid	ID	Gets or sets the unique ID for the stage. When creating a new instance, do not set; this value will be auto-generated by the database.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
string	Title	Gets or sets the title.	"Disposal Stage"
string	Description	Gets or sets the description.	"The Disposal Stage"

string	UserName	Gets or sets the UserName for who modified the entity. This will be used for logging purposes.	@"MYDOMAIN\ujohndoe"
int	Duration	Gets or sets the duration.	The valid values for duration changes according to DurationType: EeDurationType.Days: 0 - 182500 EeDurationType.Months : 0 - 6000 EeDurationType.Years: 0 - 500
EeDurationType	DurationType	Gets or sets the type of the duration.	EeDurationType.Days
List<EeEventType>	EventTypes	Gets or sets a list of the existing Event Types that is associated with this Event Stage.	new List<EeEventType>() { new EeEventType() { ID = new Guid("dacc3d47-47a6-e311-941b-000c29fe8c9f") } }
EeEventStageDateCalculationType	DateCalcType	Gets or sets the date calculation type.	EeEventStageDateCalculationType.MinOne

EeEventType

Represents a type.

Properties

Type	Name	Description	Example Values
Guid	ID	Gets or sets the unique ID for the type. When creating a new instance, do not set; this value will be auto-generated by the database.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
string	Title	Gets or sets the title.	"Employee Separation"

string	Description	Gets or sets the description.	"The Event representing the separation of an employee"
List<EeEventInstance>	Instances	Gets the instances associated with this type. Used for retrieving associated instances only. Setting this property does nothing on save.	
List<EeEventTypesAcl>	Acls	Gets the ACLs/custom permissions for this event type.	
string	UserName	Gets or sets the UserName for who modified the entity. This will be used for logging purposes.	@"MYDOMAIN\ujohndoe"
bool	UseUserAcls	Gets or sets a value indicating whether this type uses custom permissions.	
int	StageCount	Gets the number of stages to which this event type belongs.	

EeEventTypesAcl

Represents a permission.

Properties

Type	Name	Description	Example Values
Guid	ID	Gets or sets the unique ID for the permission. When creating a new instance, do not set; this value will be auto-generated by the database.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")

Guid	EventTypeID	Gets or sets the parent type's ID.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
string	EventTypeTitle	Gets the parent type's title. Setting this has no effect on saving. It is purely for information purposes.	
string	PrincipalName	Gets or sets the principal name. Should be a friendly display name.	"John Doe"
EePermissionType	PermissionSet	Gets or sets the level of permissions.	EePermissionType.TriggerEvent
string	ClaimsID	Gets or sets the claims id. See also: http://msdn.microsoft.com/en-us/library/system.security.claims.claimsidentity	"i:0#.w|mydomain\ujohndoe"
byte	AccountType	Gets or sets the type of the account.	0 // SharePoint User 1 // SharePoint group
string	SiteCollectionUrl	Gets or sets URL of the site collection for which the principal is a member.	"https://engineering.contoso.com/sites/legal"
Guid	SharePointSiteID	Gets or sets ID of the site collection for which the principal is a member.	new Guid("2c86a4f2-d9de-41c7-a358-49f750bb6233")

EePolicyMaster

Represents a policy.

Properties

Type	Name	Description	Example Values
Guid	ID	Gets or sets the unique ID for the policy. When creating a new instance, do not set; this value will be auto-generated by the database.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
string	Title	Gets or sets the title.	"Records Disposal"
string	Description	Gets or sets the description.	"Permanently Remove Records"
string	UserName	Gets or sets the UserName for who modified the entity. This will be used for logging purposes.	@"MYDOMAIN\ujohndoe"

List<EePolicyMasterStage>	Stages	Gets or sets the stages.	new List<EePolicyMasterStage>();

EePolicyMasterAssociation

Represents a target.

Properties

Type	Name	Description	Example Values
Guid	ID	Gets or sets the unique ID for the target. When creating a new instance, do not set; this value will be auto-generated by the database.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
Guid	PolicyMasterAssociationSetID	Gets or sets the parent PolicyMasterAssociationSet ID.	new Guid("e85cbafe-7ffd-48fc-9092-8c1d6e1d3864")
string	SiteUrl	Gets or sets URL of the location site.	"https://myco.hr.com/records"
Guid	SiteID	Gets or sets the ID of the location site.	new Guid("2c86a4f2-d9de-41c7-a358-49f750bb6233")
Guid	WebID	Gets or sets the ID of the location web.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
string	ContentTypeID	Gets or sets the ID of the content type.	"0x010100E1F0ECEADA9D7944AB4B2A38B6B0D85E"
Guid	LocationID	Gets or sets the ID of the location (list/folder).	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
bool	PendingDissocation	Gets or sets a value indicating if the target is pending an IMP dissociation.	false

EePolicyMasterAssociationSet

Represents an association.

Properties

Type	Name	Description	Example Values
Guid	ID	Gets or sets the unique ID for the association. When creating a new instance, do not set; this value will be auto-generated by the database.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
Guid?	NonRecordPolicyMasterID	Gets or sets the policy ID used for non-record retention schedules.	new Guid("e85cbafe-7ffd-48fc-9092-8c1d6e1d3864")
EePolicyMaster	NonRecordPolicyMaster	Gets the policy used for non-record retention schedules.	"https://myco.hr.com/records"
Guid?	RecordPolicyMasterID	Gets or sets the policy ID used for record retention schedules.	null
EePolicyMaster	RecordPolicyMaster	Gets the policy used for record retention schedules.	null
string	Title	Gets or sets the title.	"Association Set A"
string	Description	Gets or sets the description.	
List<EePolicyMasterAssociation>	Associations	Gets or sets the targets for this association.	
bool	RecordSameAsNonRecord	Gets or sets a value indicating if the record policy is the same as the non-record policy. Using the UI, the default is true.	true
bool	PendingDissocation	Gets or sets a value indicating if the association set is pending an IMP dissociation.	false

EePolicyMasterStage

Represents the stages used for a policy.

Properties

Type	Name	Description	Example Values
Guid	PolicyStageID	Gets or sets the identifier of the stage.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
int	StageNumber	Gets or sets the stage number. Used for ordering both on the UI and when creating the retention schedule.	0
string	Action	Gets or sets the action.	"Microsoft.Office.RecordsManagement.PolicyFeatures.Expiration.Action.MoveToRecycleBin"
string	ActionXml	Gets or sets the action XML.	"<action type=\"action\" id=\"Microsoft.Office.RecordsManagement.PolicyFeatures.Expiration.Action.MoveToRecycleBin\" />"
string	AdditionalData	Gets or sets the additional data. Used when the action is "StartWorkflow", "Microsoft.Office.RecordsManagement.PolicyFeatures.Expiration.Action.SubmitFileCopy", "Microsoft.Office.RecordsManagement.PolicyFeatures.Expiration.Action.SubmitFileMove", or "Microsoft.Office.RecordsManagement.PolicyFeatures.Expiration.Action.SubmitFileLink".	AdditionalData will include the Workflow name associated or Send To Location name.

EePolicyStageUsage

Only used by File Plan Builder. Do not use.

Properties

Type	Name	Description
Guid	ID	
Guid	PolicyStageID	
Guid?	FarmID	
string	LocationID	
string	ApplicationID	

EeSiteRegistration

Represents a site registration.

Properties

Type	Name	Description	Example Values
Guid	ID	Gets or sets the unique ID for the site registration. When creating a new instance, do not set; this value will be auto-generated by the database.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
string	UserName	Gets or sets the user name used in the connection credentials to the SharePoint site collection.	@"MYDOMAIN\ujohndoe"
string	Password	Gets or sets the password used in the connection credentials to the SharePoint site collection.	"kRypt0n1sH0m3"
string	Url	Gets or sets URL of the site collection.	"https://myco.hr.com/records"
string	AdfsUrl	Gets or sets URL of the ADFS server. Only used if the registration is an ADFS site.	"https://adfs.myco.hr.com/records"
short	RegistrationType	Gets or sets the type of the registration.	0 // SharePoint2013KerbNtlm, 1 // SharePoint2013Adfs,

			2 // SharePoint2010KerbNtlm, 3 // SharePoint2010Adfs, 4 // Office365 (Not currently used)
Guid	SharePointSiteID	Gets or sets the SharePoint site collection ID.	new Guid("84045690-9451-43bb-a29a-8ea3025d4b18")
Guid?	SharePointFarmID	Gets or sets the SharePoint Farm ID.	

ValidationError

Properties

Type	Name	Description
string	ErrorMessage	The error message associated with the validation.
IEnumerable<string>	MemberNames	The name of the members that have validation errors.

Enums

EeDurationType

public enum EeDurationType : short { Days, Months, Years }

EeEventStageDateCalculationType

public enum EeEventStageDateCalculationType : byte { MinOne = 0, MinAll = 1, MaxOne = 2, MaxAll = 3 }

EePermissionType

public enum EePermissionType { NoAccess = 0, ReadEvent = 1, UpdateEvent = 2, DeleteEvent = 4, TriggerEvent = 8, ManagePermissions = 16, FullControl = 32 }

Operator

public enum Operator { EqualTo = 0, NotEqualTo = 1, GreaterThan = 2, GreaterThanOrEqualTo = 3, LessThan = 4, LessThanOrEqualTo = 5, IsBetweenInclusive = 6, IsBetweenExclusive = 7, StartsWith = 8, EndsWith = 9, Contains = 10, IsBlank = 11, IsNotBlank = 12, EqualsTrue = 13, EqualsFalse = 14 }

Validation

EeEventAdmin

Property	Validators
PrincipalName	Required StringLength(1024)
ClaimsID	Required StringLength(2048)
SiteCollectionUrl	Required StringLength(2048)

EeEventInstance

Property	Validators
Title	Required StringLength(255) HasValidTitle – The title must be unique within an event type.
Priority	Required Range(1, 9)
UserName	StringLength(1024)
Fields	CollectionRequired(0) – *Can't be null.*

EeEventStage

Property	Validators
Title	Required StringLength(255) HasValidTitle – The title must be unique within all of enterprise events.
UserName	StringLength(1024)
Duration	Required ConditionalRange("DurationType", EeDurationType.Days, 0, 182500) ConditionalRange("DurationType", EeDurationType.Months, 0, 6000) ConditionalRange("DurationType", EeDurationType.Years, 0, 500)
EventTypes	Required CollectionRequired – Can't be null and must have at least one item.
DateCalcType	Required

EeEventType

Property	Validators
Title	Required StringLength(255) HasValidTitle – The title must be unique within all of enterprise events.
UserName	StringLength(1024)

EeEventTypesAcl

Property	Validators
EventTypeID	Required
PrincipalName	Required StringLength(1024)
PermissionSet	Required EqualTo – Only accepted values are NoAccess, TriggerEvent, or FullControl.
ClaimsID	Required StringLength(2048)
SiteCollectionUrl	Required StringLength(2048)
AccountType	Required
SharePointSiteID	Required

EePolicyMasterAssociation

Class level validator: IsNewAssociation. Only one association can exist in Enterprise Events at a time. This validator is a duplicate-preventer.

EePolicyMasterAssociationSet

Property	Validators
Title	Required StringLength(255) HasValidTitle – The title must be unique within all of enterprise events.
UserName	StringLength(1024)
NonRecordPolicyMasterID	Required
RecordPolicyMasterID	RequiredIfEqual – Only required if RecordSameAsNonRecord is false.
Associations	CollectionRequired – Can't be null and must have at least one item.

EePolicyMaster

Property	Validators
Title	Required StringLength(255) HasValidTitle – The title must be unique within all of enterprise events.
UserName	StringLength(1024)
Stages	Required CollectionRequired – Can't be null and must have at least one item.

EeSiteRegistration

Class level validator: HasValidSharePointSiteID. Validates the Url by attempting communication with SharePoint. Could potentially fail on invalid credentials or inability to connect. Also verifies that it's a Site Collection URL, and not a Web URL. Also verifies that the Registration Type matches the SharePoint URL (i.e., make sure that you chose SharePoint 2013 and the Url is SharePoint 2013). Finally, ensures only one site registration with the returned Site Collection ID.

Property	Validators
UserName	Required StringLength(1024)
Password	Required
Url	Required MaxLength(2048) GimmalSoft.EnterpriseEvents.Web.UrlAttribute
AdfsUrl	MaxLength(2048) GimmalSoft.EnterpriseEvents.Web.UrlAttribute RequiredIfEqual – Only required if RegistrationType is ADFS.

	CustomValidation("IsValidAdfsUrl") – Validates the ability to get an auth cookie from the ADFS server.
RegistrationType	Required