API Enhancements in Vault 2019

Paul Gunn
Vault Software Architect
Paul Gunn

Paul has been a member of the Vault team since 2003. Throughout that time, he has been involved in the development of many features of the product including security, search, and replication with a focus on server-side functionality. Paul currently serves as a software architect for Vault.
Learning Objectives

• Learn about new API functionality available in Vault 2019

• Discover how this functionality was used to implement Project Sync

• Understand how these capabilities can be used in a custom application

• See the code behind and understand how it works
What is Project Sync?
How is sync configuration stored?
Entity attributes overview

- Entity Attributes allow data to be programmatically associated with any entity.
- This data is not directly user-visible and can be applied to read-only entities.
- These are not considered part of history so do not create new file versions.
- Attributes are in a user-defined namespace so multiple usages don't collide.
Entity attribute object (EntAttr)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attr</td>
<td>User-defined name of the attribute.</td>
</tr>
<tr>
<td>Cloaked</td>
<td>Is the entity cloaked for the current user.</td>
</tr>
<tr>
<td>EntityId</td>
<td>Entity tagged with this attribute.</td>
</tr>
<tr>
<td>Val</td>
<td>Value of the attribute.</td>
</tr>
</tbody>
</table>
Entity attributes API

- `void SetEntityAttribute(long entityId, string namespc, string attribute, string val)`
  - Sets a named attribute on a given entity. A null val will delete an existing attribute.
- `EntAttr[] GetEntityAttributes(long entityId, string namespc)`
  - Gets all entity attributes associated with a given entity.
- `EntAttr[] FindEntityAttributes(string namespc, string attribute);`
  - Finds all entity attributes with a given attribute name.
- `EntAttr[] FindAllEntityAttributes(string namespc)`
  - Finds all entity attributes in the given namespace.
How are sync jobs scheduled?
Scheduled jobs overview

• Legacy functionality supports adding job for one-time, immediate execution
• New functionality supports scheduling a recurring job to run at a given cadence
• At scheduled times, a traditional job is added to the queue for normal execution / consumption
• Scheduled jobs can be viewed with other background tasks in ADMS console
### Scheduled job object (SchedJob)

<table>
<thead>
<tr>
<th><strong>Field</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateDate</td>
<td>The date the job was created.</td>
</tr>
<tr>
<td>CreateUser Id</td>
<td>The ID of the user who created the job.</td>
</tr>
<tr>
<td>CreateUser Name</td>
<td>The name of the user who created the job.</td>
</tr>
<tr>
<td>Descr</td>
<td>A description of the job.</td>
</tr>
<tr>
<td>ExecDate</td>
<td>DateTime at which the job is first scheduled (can be DateTime.Now).</td>
</tr>
<tr>
<td>ExecFreq</td>
<td>Frequency in minutes at which to schedule the job (e.g. 1440 minutes = daily).</td>
</tr>
<tr>
<td>Id</td>
<td>A unique identifier for the job.</td>
</tr>
<tr>
<td>IsOnSite</td>
<td>In a multi-site environment, this property tells if the file is on the local site.</td>
</tr>
<tr>
<td>ParamArray</td>
<td>An array of parameters which provide meta-data about the job.</td>
</tr>
<tr>
<td>Priority</td>
<td>The priority of the job. A lower number means a higher priority. 1 is the lowest possible number.</td>
</tr>
<tr>
<td>Typ</td>
<td>The job type.</td>
</tr>
<tr>
<td>VaultId</td>
<td>The ID of the Vault that the job applies to.</td>
</tr>
</tbody>
</table>
Scheduled job API

- SchedJob AddScheduledJob(string type, string desc, JobParam[] paramArray, int priority, System.DateTime execDate, int execFreqInMinutes)
  - Adds a scheduled job with given execution date and frequency
- void DeleteScheduledJob(long id)
  - Deletes the given scheduled job
- SchedJob GetScheduledJob(long id)
  - Gets information about the given scheduled job
- SchedJob[] GetScheduledJobs()
  - Gets information about all scheduled jobs
Vault Notification Sample
Demo
Limitations of the Sample Application

• Email
  o Assumes the email SMTP server is on ‘localhost’
  o Assumes the email addresses for vault user have been correctly configured

• Job Scheduling
  o A different scheduled job is created for each user, which could be inefficient
  o The user must have permissions to create a scheduled job

• Fit and finish
  o There are no options for including file dependencies and drawings
  o Integration with Vault Explorer via command extension
How is the notification list stored?
Design of the notification list

```csharp
private string AttributeNamespace
{
    get { return "Sample.VaultNotification." + UserName; }
}

private const string MostRecentAttribute = "MostRecentIterationId";
```
Loading the notification list

```csharp
public IEnumerable<IEntity> Load()
{
    var attributes = Connection.WebServiceManager.PropertyService.FindEntityAttributes
        (AttributeNameSpace, MostRecentAttribute) ?? Enumerable.Empty<ACW.EntAttr>();

    attributes = attributes.Where(a => !a.Cloaked); // ignore files we no longer have access to

    if (!attributes.Any())
        return Enumerable.Empty<IEntity>();

    var resolvedIds = Connection.PersistableIdManager.ResolvePersistableIds(attributes.Select(a => a.Val));
    return resolvedIds.Select(rid => rid.Value);
}
```
Modifying the notification list

```csharp
public void Add(IEnumerable<IEntity> entities)
{
    var persistentIds = Connection.PersistenceIdManager.GetPersistableIds(entities, getLatest: false);

    foreach (var current in persistentIds)
    {
        Connection.WebServiceManager.PropertyService.SetEntityAttribute
            (current.Key.EntityMasterId, AttributeNamespace, MostRecentAttribute, current.Value);
    }
}

public void Remove(IEnumerable<IEntity> entities)
{
    foreach (var ent in entities)
    {
        Connection.WebServiceManager.PropertyService.SetEntityAttribute
            (ent.EntityMasterId, AttributeNamespace, MostRecentAttribute, null);
    }
}
```
Updating the notification list

```csharp
public IEnumerable<IEntity> Update(IEnumerable<IEntity> entities)
{
    var modified = GetModified(entities);
    if (!modified.Any())
        return Enumerable.Empty<IEntity>();

    var updatedFiles = Connection.FileManager.GetLatestFilesByIterationIds(
        modified.Select(e => e.EntityIterationId)).Values;

    Add(updatedFiles);
    return updatedFiles;
}
```
How is the notification job work?
public void Create()
{
    if (m_job != null)
        return;

    // Schedule job recurring every 24 hours at midnight.
    //
    var frequency = TimeSpan.FromDays(1);
    var timeofday = DateTime.ParseExact("00:00", "HH:mm", System.Globalization.CultureInfo.InvariantCulture);
    var param = new ACW.JobParam() { Name = NotificationJobUserName, Val = m_conn.UserName };

    m_job = m_conn.WebServiceManager.JobService.AddScheduledJob
        (NotificationJobType, "Watch list notifications for " + m_conn.UserName, new [] {param},
         50, timeofday, (int)frequency.TotalMinutes);
}
Executing the notification job

```csharp
public JobOutcome Execute(IJobProcessorServices context, IJob job)
{
    var userName = job.Params[NotificationJob.NotificationJobUser];
    if (userName == null)
    {
        context.Log("User name parameter was not specified on the job", MessageType.eError);
        return JobOutcome.Failure;
    }

    var notificationList = new NotificationList(context.Connection, userName);
    var notificationReport = new NotificationReport(context.Connection, notificationList);
    notificationReport.SendReport();
    notificationList.Update(notificationReport.Modifications);

    return JobOutcome.Success;
}
```
Conclusion
Call to action

- Project Sync was built on top of these API foundations
- Other future features are also being built on this functionality
- Vault has no secret server APIs: what we can do, you can do.
- Amaze us with what you can do!