Scripting

API Essentials

Key Entity Interactions

The following API techniques can be used on nearly every configuration and data entity in the TrakSYS database. It is essential to master and use these basics when scripting in TrakSYS...

Load

Load a single instance of an entity from the database to an API model class for use.

Get List

Load a List of entities from the database to an API model class for use.

Insert

Insert a new entity into the database from a populated API model class.

Update

Update a modified entity into the database from a populated API model class.

Delete

Remove an entity from the database by its unique identifier.

Scripting

API Essentials | Load

Load by ID

This method loads a TrakSYS entity from the database based on the id argument passed in.

```
// create a model object to hold the results of the load
ETS.Core.Api.Models.Data.DbEvent ev;

// load the entity with ID 123 from the database
ev = api.Data.DbEvent.Load.ByID(123);

// access the properties of the model object
DateTime startDateTime = ev.StartDateTime;
int eventDefinitionID = ev.EventDefinitionID;
// etc...
```

Load with SQL

This method loads a TrakSYS entity from the database based on the sql string argument passed in. The SQL specified must return ALL the fields from the appropriate entity's database table.

```
// create a model object to hold the results of the load
ETS.Core.Api.Models.Data.DbEvent ev;

// load the first entity from the database
ev = api.Data.DbEvent.Load.WithSql("SELECT TOP 1 * FROM tEvent");

// access the properties of the model object
DateTime startDateTime = ev.StartDateTime;
int eventDefinitionID = ev.EventDefinitionID;
// etc...
```

Scripting

API Essentials | Get List

Get List with SQL

This method loads a List of TrakSYS entities from the database based on the sql string argument passed in. The SQL specified must return ALL the fields from the appropriate entity's database table.

```
// declare list object
List<DbMaterial> materials = new List<DbMaterial>();

// load list
materials = api.Data.DbMaterial.GetList
      .WithSql("SELECT * FROM tMaterial");

// loop through the list
foreach (DbMaterial material in materials)
{
    // use material object
}
```

Get List by FK

There are several methods used to retrieve Lists of Model objects based on the database entity's typical foreign keys. The type of Model object retrieved is based on the parent service name. For example, Api.Data.ListOf.DbAreas.GetList will return a List of DbArea Model objects (List<DbArea>).

```
// get a List of DbSystems given a specific Area ID (3)
List<DbSystem> systems =
      api.Data.DbSystem.GetList.ByAreaID(3);

// get a List of DbSystems given a specific SystemTypeID (5)
List<DbSystem> systems =
      api.Data.DbSystem.GetList.BySystemTypeID(5);
```

Scripting

API Essentials | Insert and Update

Insert

This method inserts a TrakSYS entity passed in as the item argument, to the database.

```
// create a model object to populate
ETS.Core.Api.Models.Data.DbEvent ev = new
ETS.Core.Api.Models.Data.DbEvent();

// populate the model object as needed
ev.StartDateTime = api.Site.GetCurrentDateTime();
ev.EventDefinitionID = 45;

// create a result object to determine the success of the operation
ETS.Core.Api.Models.Result<ETS.Core.Api.Models.Data.DbEvent> result;

// insert the entity in the database
// (the ID will be created automatically upon insert)
result = api.Data.DbEvent.Save.InsertAsNew(ev);

// examine the results of the operation
if (result.Success) { int newID = result.Return.ID; }
else { // failure code }
```

Update

This method updates an existing TrakSYS entity passed in as the item argument, to the database.

```
// create a model object to hold the results of the load
ETS.Core.Api.Models.Data.DbEvent ev;

// load the entity with ID 123 from the database
ev = api.Data.DbEvent.Load.ByID(123);

// modify the properties of the model object as needed
ev.Notes = "new notes have been added";

// create a result object to determine the success of the operation
ETS.Core.Api.Models.Result<ETS.Core.Api.Models.Data.DbEvent> result;

// update the entity in the database
result = api.Data.DbEvent.Save.UpdateExisting(ev);

// examine the results of the operation
if (result.Success) { // success code }
else { // failure code }
```

Scripting

API Essentials | Unit of Work

API Transaction Mechanism

The UnitOfWork object represents a coding pattern, that in this instance is used to represent a database transaction.

```
// setup api object
var api = ApiService.GetInstance();

// create UnitOfWork
var uow = api.CreateUnitOfWork();

// create parent area
var area1 = api.Data.DbArea.Create.FromParentNone();
area1.Name = "Test";
area1 = api.Data.DbArea.Save.InsertAsNew(area1, uow).Return;
// does not save to database yet, but does populate ID and DisplayOrder

// creates a child area
var area2 = api.Data.DbArea.Create.FromParentNone();
area2.Name = "Child Area to Test";
area2.ParentAreaID = area1.ID; // assigns parentArea ID to area2
area2 = api.Data.DbArea.Save.InsertAsNew(area2, uow).Return;
// does not save to database yet

// now saves both areas to the database
uow.Execute();
```

Scripting

Extension Methods

The TrakSYS API provides many extension methods that extend and enable useful functionality when scripting.

Color Extensions

These additional methods are available from any Color or string object...

Color.ToCssStyle
Color.ToHexDisplay
Color.Lighted
Color.Darken
string.AsColor

For example...

```
Color c = Color.Blue;
string s = c.ToCssStyle();
s = "red";
c = s.AsColor();
```

TsColor Object

Use the TsColor data type to easily create and convert from the TrakSYS named colors...

```
string s = TsColor.Success.ToCssClass();
```

Dictionary Extensions

These additional methods are available from any IDictionary object (like **this.Ets.Values**) and assist with retrieving items into specific data typed variables...

GetAsBool
GetAsDateTime
GetAsDateTimeOffset
GetAsDouble
GetAsInt
GetAsString
RemoveByKeyIfExists

For example...

```
int i = this.Ets.Values.GetAsInt("SystemID", -1);
```

String Conversion Extensions

These additional methods are available from any string variable and assist with converting to different data types...

AsBool
AsDateTime
AsDateTimeOffset
AsDouble
AsEnum
AsInt

For example...

```
string s = "1234";
int i = s.AsInt();
```

Variant String Conversion Extensions

These additional methods are available from any string variable. The conversions are done using special assumptions about how decimals and group separators are stored in the TrakSYS variant format (always a period and never any group separators)...

AsBoolFromVariant
AsDoubleFromVariant
AsIntFromVariant

For example...

```
var job = this.Ets.Api.Data.DbJob
      .Load.ByID(23)
      .ThrowIfLoadNull("Error Loading Job!");
// temperature is in Capture01
double d =
job.Capture01.AsDoubleFromVariant(0);
```

String SQL Extension

This additional method is available from any string variable and is used to properly encode data into SQL statements...

```
var sys = this.Ets.Api.Data.DbSystem.Load
      .WithSql(
      "SELECT * FROM tSystem WHERE Key = {0}"
      .FormatWith("L1".ToSql()));
```

Error Handling

The TrakSYS API exposes several extension methods to assist with more compact and consistent exception handling.

- Must be used within a Try / Catch
- Content Page/Part Methods auto Try / Catch

ThrowIfNull

Available for any method that returns an object. An exception will be thrown if the return is **null**. A custom error message can be provided.

```
// GetList.ForParentAreaID returns null if there is a failure
var areas = this.Ets.Api.Data.DbArea.GetList.ForParentAreaID(6)
      .ThrowIfNull("Error loading for Parent Area 6!");
```

ThrowIfLoadFailed

Available for any method that returns an object. An exception will be thrown if the return is **null**. Ideal for API Load methods where a consistent error message is desired based on the property being loaded by.

```
// Load.ByID returns null if there is a failure
var area = this.Ets.Api.Data.DbArea.Load.ByID(3)
      .ThrowIfLoadFailed("ID", 3);
// exception error message = Unable to load DbArea with ID 3
```

ThrowIfFailed

Available for any method that returns a TrakSYS **Result<T>** object. An exception will be thrown if the **Result.Success** value is **false**. . A custom error message can be provided.

```
// Save.UpdateExisting returns a Result<DbArea> object
this.Ets.Api.Data.DbArea.Save.UpdateExisting(area)
      .ThrowIfFailed("Error updating Area!");
```

Scripting

Result Object

The TrakSYS API exposes and makes use of a **Result<T>** return object that allows the return of both a **Success** Boolean indicating the success of the operation, and a **Return** object specified by **T**.

- Most TrakSYS API methods use Return<T>
- .Success | True or False
- .Return | object of type T
- .Messages | Errors if Success = False
- Many TrakSYS API Uses
- Recommended as return for new Methods
- If a method returns a Result object, use it!

```
// get Result from Save.InsertAsNew method
var res = this.Ets.Api.Data.DbArea.Save.InsertAsNew(newArea);

// examine Result.Success
if (res.Success)
{
      // use Result.Return
      var retArea = res.Return;
      int newID = retArea.ID;
}
else
{
      // log messages
      this.Ets.Debug.FailFromResultMessages(res.Messages);
}
```