Hitachi Solutions

Hsl.OrgService Functionality in Hsl JavaScript Library v9

API Reference

Aug 25, 2020

Contents

SOAP Endpoint (library.orgservice.js)

To use the functionality described in this part of the document, the library.orgservice.js web resource must be added to the form.

Hsl.OrgService (namespace)

The OrgService namespace contains methods for calling CRM services and methods return promise objects rather than using standard callbacks.

The ajax methods all take a ServiceOptions object which allows configuring how the ajax executes.

Service calls in this namespace generally return Promises. The fail callback for these promises passes an error object with the properties described in the SoapError section.

create

Create an entity record. Promise callback has an Id attribute that gives the id of the created record.

```
Hsl.OrgService.create(entity: Entity, opts?: ServiceOptions): Promise<CreateResponse>;
```

```
var account = Hsl.entity('account');
account.set('name', 'My Test Account');
account.setOptionSetValue('preferredcontactmethodcode', 2); // Email
Hsl.OrgService.create(account).then(function (res) {
    Hsl.openEntityForm('account', res.Id, null, {
        openInNewWindow: true
    });
}, function (err) {
    Hsl.Dialog.showError(err, 'Error creating account:');
});
```

update

Updates an entity record.

```
Hsl.OrgService.update(entity: Entity, opts?: ServiceOptions): Promise<{}>;
```

```
var account = Hsl.entity('account', Xrm.Page.data.entity.getId());
account.set('name', 'My Test Account');
account.setOptionSetValue('preferredcontactmethodcode', 2); // Email
Hsl.OrgService.update(account).then(function (res) {
    Hsl.openEntityForm('account', res.id, null, {
        openInNewWindow: true
    });
}, function (err) {
    Hsl.Dialog.showError(err, 'Error updating account:');
});
```

erase

Deletes the entity record. Delete is a reserved word in JavaScript so erase was used instead.

```
Hsl.OrgService.erase(entity: EntityReference, opts?: ServiceOptions): Promise<{}>;
```

retrieve

Retrieves a record. When specifying columns, be sure to list as few as necessary for best performance.

```
Hsl.OrgService.retrieve(entity: EntityReference, columns: string[], opts?: ServiceOptions): Promise<RetrieveResponse>;
```

The response has the following field:

- Entity – EntityClass – the record retrieved.

fetch

Executes a fetch xml query. Note: fetch queries will only return the smaller of the “count” attribute or 5,000 records. To get the next page of results, use fetchNextPage.

```
Hsl.OrgService.fetch(fetchXml: string, opts?: ServiceOptions): Promise<ExecuteFetchResponse>;
```

The response has the following fields:

- EntityCollection: EntityCollectionClasss – the query results.
- Entities: EntityClass[] – shortcut to EntityCollectionClass.Entities.

fetchNextPage

Retrieves the next page of results for the given query.

```
Hsl.OrgService.fetchNextPage(resp: ExecuteFetchResponse>, opts?: ServiceOptions): Promise<ExecuteFetchResponse>;
```

The response will have the same fields as `fetch`’s response object.

```
Hsl.OrgService.fetch(fetchQuery).then(function (resp) {
   var results = resp.Entities;
   if (resp.EntityCollection.MoreRecords) {
       return Hsl.OrgService.fetchNextPage(resp).then(function (nextPageResp) {
           // Process next page
       });
   }
});
```

setState

Executes a fetch xml query.

```
Hsl.OrgService.setState(entity: EntityReference, state: number, status?: number, opts?: ServiceOptions):
        Promise<OrganizationResponseClass>;
```

The response has the following fields:

- EntityCollection: EntityCollectionClasss – the query results.
- Entities: EntityClass[] – shortcut to EntityCollectionClass.Entities.

execute

Executes an organization request. See Hsl.organizationRequest. The response will be an OrganizationResponse object with properties for `xdoc` and `results`. The `results` property is a key/value dictionary. The JS Library sample code has several examples of executing organization requests.

```
Hsl.OrgService.execute(req: OrganizationRequestClass, opts?: ServiceOptions): Promise<OrganizationResponseClass>;
```

```
var req = Hsl.organizationRequest('WhoAmI');
Hsl.execute(req).then(function (response) {
   // Get response output from response.results or parse out response with response.xdoc.
   Hsl.Dialog.alert(response.results.userId);
}, Hsl.Dialog.showError);
```

executeService

Returns a deferred. Execute Service calls a service writing using the Hitachi Solutions Service plugin template. The object passed in has the following interface:

```
Hsl.OrgService.executeService(settings): Promise<ServiceResponse>;
```

The three modes use different methods for calling the service:

- Action
    - Registered to Post Operation of Custom Action, Synchronous
    - Uses a custom action. The serviceName should be the name of the custom action.
    - The custom action should have parameters:
        - Name – string, input
        - InData – string, input

        - OutData – string, output
    - Atomic if you enable rollbacks on the custom action configuration
- Create
    - Plugin registered to pre-operation of Create Synchronous or post-operation of Create Asynchronous.
    - Atomic if registered synchronously
    - Creates a custom entity to execute the request. The entity should have the following attributes (case sensitive) with the same customization prefix as the custom entity. For example, I could create a custom entity named hsl_mytestrequest with the following attributes: hsl_name, hsl_Data, hsl_IsComplete, and hsl_Progress.
        - name (Single Line of Text)
        - Data (Multiple Lines of Text)
        - IsComplete (Two Options)
        - Progress (Whole Number or String)
- RetrieveMultiple
    - Plugin Registered to Post Operation of RetreiveMultiple, synchronous.
    - Not atomic
    - Executes a RetrieveMultiple for the custom entity. The entity should have the following attributes with the same customization prefix as the custom entity.
        - Name
        - Data

The sample code in ExecuteService.js contains samples of the execute service call.

`ServiceResponse` has a field `data` that contains the deserialized response from the server.

```
/** Name of the entity or custom action to execute. When passing in a name parameter, add a dot followed by the name of
the operation.  */
serviceName: string;
/** Data to pass to the service. */
data?: any;

/** Method to use to call the service. "Action", "Create", or "Retrieve".
Default: "Action" */
method?: string;

/** Whether to make the AJAX call async. Default: true */
async?: boolean;

/** Used with async create. This will be called any time the created record’s progress field changes. */
progress?(obj: {
    /* The progress value from the CRM record. The data type will depend on the data type in CRM. */
    progress: any;
    /** Data is only passed to the progress callback during the first callback execution. After that, it will be null. */
    data?: any;
}): void;

/** Polling frequency for async create in milliseconds. */
pollingFrequency?: number;
/** Data type of response. Either JSON or string Default: JSON. */ responseType?: string;
/** Service call timeout in milliseconds */
timeout?: number;
```

executeTransaction

CRM Online has a limit of 2 concurrent batch requests at a time. Any subsequent requests that come in before the first two finish will get a "Server Busy" error. As such, this method generally isn't suitable for frequent use with CRM Online deployments.

As an alternative, you can use a custom action to implement a transaction by checking the "Enable Rollback" button.

Executes a set of requests in a single database transaction. Be careful when using this request – long running transactions will lock records and can cause slowness for other users. The options parameter has all the same settings as ServiceOptions except it has an additional setting `returnResponses`. When true, the responses for each request will be returned as an array on the `responses` property of the response. The responses will be in the same order as the requests.

```
Hsl.OrgService.executeTransaction(requests: Hsl.OrganizationRequestClass[], opts?: ExecuteTransactionOptions):
        Promise<ExecuteTransactionResponse>;
```

```
// Delete the current record then set create a new record with a status of "Started" and open it.
// If the delete or create fails, the transaction will be rolled back.
var deleteRequest = Hsl.organizationRequest('Delete');
deleteRequest.setEntityReference('Target', Hsl.Form.currentRecord());

var createObj = Hsl.entity('account');
createObj.setOptionSetValue('statusreason', 1 /* Started */);
var createRequest = Hsl.organizationRequest('Create');
createRequest.setEntity('Target', createObj);

Hsl.OrgService.executeTransaction([deleteRequest, createRequest], { returnResponses: true }).then(function (resp) {
        var createResponse = resp.responses[1]; // Responses will have the same index as the matching request.
        Hsl.openEntityForm('hsl_testentity', createResponse.get('id'));
}, function (err) {
   Hsl.Dialog.showError(err);
});
```

ServiceOptions (Interface)

Methods is Hsl.OrgService accept ServiceOptions as their final parameter. ServiceOptions has the following fields:

- **async?: boolean** – [DEPRECATED] whether the ajax call will be made synchronously or asynchronously. Default: true
  Note: if executing synchronous ajax, using the promise returned from the method call is deprecated. Use the success/fail/always settings instead.
  Note: it is best to avoid synchronous ajax. It is deprecated and may be removed in future versions of browsers. Synchronous AJAX results in a poor user experience where the browser is frozen while the call is being made.
- **timeout?: number** – Timeout for the service call in milliseconds. (Not available on synchronous requests) Default: 30000
- **showBusy?: boolean** – If true, a busy spinner will appear on the page preventing the user from interacting with the form.
- **preventSave?: boolean** – If true, the user will be unable to save the form while the ajax call is in progress. You may want to use this setting in scenarios like defaulting a field's value.
- **success?: (t: any) => void** – Called after the ajax operation completes successfully.
- **fail?: (error: SoapError) => void** – Called after the ajax operation fails.
- always?: (t: any, error: SoapError) => void; - Called after the ajax operation completes regardless of success or failure. Called after the success/fail callbacks have completed.

SoapError

Rejections from the SOAP endpoint generally have each of the properties listed below. The method Hsl.Dialog.showError is a helper method designed specifically to display these errors. It will display the fault text with the full message collapsed by default.

- errorCode – Integer error code returned from the service, if available. See http://msdn.microsoft.com/en-us/library/gg328182.aspx for reference.
- faultMessage – The fault error message
- message – The error message
- detailedMessage – Details message of the error.
- traceText – Trace text from the plugin returning the error if applicable
- requestText – The request text
- responseText – The response body text, if available.

Hsl.organizationRequest

Helper class for executing CRM Service requests.

Creation

Creates a new organization request instance for the given request name. Pass in the request name without the "Request" postfix. So Hsl.organizationRequest("Recalculate") is correct while ~~Hsl.organizationRequest("RecalculateRequest")~~ is incorrect. Once created, the request can have parameters set using the various set methods below then be executed using Hsl.OrgService.executeOrganizationRequest.

```
Hsl.organizationRequest(requestName: string): OrganizationRequest;
```

Set Methods

Set parameters on the organization request using the methods. For custom actions, refer to the action. For other request types, refer to the data types for those requests.

For example, the documentation for InstantiateTemplate states that the input parameters are ObjectId: Guid, ObjectType: string, and TemplateId: Guid so you'd want to set those three parameters with the associated datatypes for executing that request.

Dynamics CRM SDK Request Reference:
http://msdn.microsoft.com/en-us/library/microsoft.crm.sdk.messages.aspx
http://msdn.microsoft.com/en-us/library/microsoft.xrm.sdk.messages.aspx

```
setString(key: string, val: string): void;
setDateTime(key: string, val: Date): void;
setDateTime(key: string, val: string): void;
setDecimal(key: string, val: number): void;
setMoney(key: string, val: number): void;
setInt(key: string, val: number): void;
setFloat(key: string, val: number): void;
setBoolean(key: string, val: boolean): void;
setGuid(key: string, val: string): void;
setEntityReference(key: string, val: EntityReference): void;
setOptionSetValue(key: string, val: number): void;
setEntity(key: string, val: EntityClass): void;
setEntityCollection(key: string, val: EntityCollectionClass): void;
setOptionSetValue(key: string, val: number): void;
/* Null can be passed as the value for any of the other set methods but setNull
 * can be handy if you don't know the data type.*/
setNull(key: string): void;
setEntityReferenceCollection(key: string, val: EntityReference[]): void;
setRelationship(key: string, val: { SchemaName: string; }): void;
/* Used to set binary data for requests that take a byte[] parameter. */
setBinary(key: string, base64: string): void;// Data should be base64 encoded.
/* Used to set a fetch query for requests that take a QueryBase or FetchExpression
 * parameter like BulkDetectDuplicatesRequest.
 * Note: at this time, the only option for setting QueryBase parameters is FetchExpression.
 *       Serializers for QueryExpression and QueryByAttribute are not implemented. */
setFetchExpression(key: string, fetchXml: string): void;
```

Hsl.entity

Hsl.entity (EntityClass) creates a new entity object for use with `Hsl.OrgService`.

```
Hsl.entity(logicalName: string, id?: string): EntityClass;
```

Note, that this doesn't retrieve the record in question but is meant to be used with methods like `Hsl.OrgService.create` or `Hsl.OrgService.update`.

Id

This property gets or sets the guid primary id of the record.

```
entity.Id = '{2E5F86A4-DB8D-4D00-80BE-DF2105D3AB7D}';
```

```
entity.Id: string;
```

LogicalName

This property gets or sets the logical name of the entity.

```
var isTeam = entity.LogicalName === 'team';
```

```
entity.LogicalName: string;
```

contains

Returns whether the entity contains a value for the given key.

```
entity.contains(key: string): boolean;
```

getKeys

Returns that keys for all attributes that the entity contains.

```
entity.getKeys(): string[];
```

remove

Removes a key from the entity.

```
entity.remove(key: string): boolean;
```

clone

Creates a clone of this entity with the same Id, LogicalName, Formatted Values, and Attribute Values.

```
entity.clone(): Hsl.entity;
```

Getter methods

get

Returns one of the following types based on the data type of the value:

- null
- string (String, memo, and guid fields)
- number (decimal, double, integer, money, and option set fields)
- boolean (boolean fields)
- Date (Date time fields)
- EntityReference (Lookup fields) (See EntityReference)
- EntityCollectionClass (Party list fields)

```
entity.get(key: string): any;
```

getPartyList

A shortcut for getting the party ids from an EntityCollection on a party list field.

```
entity.getPartyList(key: string): EntityReference[];
```

getRefId [Deprecated]

Deprecated: Use getValueId instead.

Gets the id from an entity reference attribute.

```
entity.getRefId(key: string): string;
```

getRefName [Deprecated]

Deprecated: Use getValueName instead.

Gets the display name from an entity reference attribute.

```
entity.getRefName(key: string): string;
```

getRefLogicalName [Deprecated]

Deprecated: Use getValueEntityName instead.

Gets the entity logical name from an entity reference attribute.

```
entity.getRefLogicalName(key: string): string;
```

getValueId

Gets the id from an entity reference attribute.

```
entity.getRefId(key: string): string;
```

getValueName

Gets the display name from an entity reference attribute.

```
entity.getRefName(key: string): string;
```

getValueEntityName

Gets the entity logical name from an entity reference attribute.

```
entity.getRefLogicalName(key: string): string;
```

getFormattedValue

Returns the value formatted based on the current users settings.

```
entity.getFormattedValue(key: string): string;
```

getDisplayString

Returns the formatted value if it is available. Otherwise, returns the record name or string representation of the value.

```
entity.getDisplayString(key: string): string;
```

getDataType

Returns the data type used by the library. This method is best used with the generic “set” method when working with generic data. For example, if `source` were an `Hsl.entity` object below in the sample below, you could copy fields generically using getDataType.

```
// Copy field values from source to target.
var fieldsToCopy = ['hsl_name', 'hsl_datefield', 'hsl_lookupfield'];
var target = Hsl.entity('hsl_entityname');
fieldsToCopy.forEach(function (attr) {
  var value = source.get(attr);
  var type = source.getDataType(attr);
  target.set(attr, value, type);
});
```

```
entity.getDataType(key: string): string;
```

Setter Methods

Set attributes on the entity using the methods below.

```
setString(key: string, val: string): void;
setDateTime(key: string, val: Date): void;
setDateTime(key: string, val: string): void;
setDecimal(key: string, val: number): void;
setMoney(key: string, val: number): void;
setInt(key: string, val: number): void;
setFloat(key: string, val: number): void;
setBoolean(key: string, val: boolean): void;
// setGuid is only used by a few special fields like processid/stageid on BPF enabled entities.
// For any lookup field, you should use setEntityReference instead. You can see each attributes
// data type in the metadata.
setGuid(key: string, val: string): void;
setEntityReference(key: string, val: EntityReference): void;
setOptionSetValue(key: string, val: number): void;
setEntity(key: string, val: EntityClass): void;
setEntityCollection(key: string, val: EntityCollectionClass): void;
setOptionSetValue(key: string, val: number): void;
/* Null can be passed as the value for any of the other set methods but setNull can be handy if you don't know the data
type.*/
setNull(key: string): void;
/* Shortcut method for setting party list field values. */
setPartyList(key: string, partyList: EntityReference[]): void;
```

Hsl.entityCollection

Create an entity collection object. Can be used with `Hsl.entity` for partylist fields or with `Hsl.organizationRequest` to pass in data sets. The EntityCollectionClass data type is also used with many responses.

```
Hsl.entityCollection(): EntityCollectionClass;
```

```
var request = Hsl.organizationRequest('Validate');
var ec = Hsl.entityCollection();
ec.Entities.push(activity);
request.setEntityCollection('Activities', ec);
Hsl.OrgService.execute(request);
```

EntityCollection objects have the following definition:

Property	Type	Notes
Entities	Hsl.entity[]	
MoreRecords	Boolean	
EntityName	String	
PagingCookie	String	
TotalRecordCount	Number	
TotalRecordCountLimitExceeded	Boolean	

Hsl.Metadata (namespace)

The methods in this namespace provide helpers for retrieving entity metadata.

The data model for EntityMetadata isn't in this document but you can find it in HslCrmLibrary.d.ts which contains TypeScript definitions. Even if you are writing normal JavaScript, you can still reference the metadata type definitions to help write code that uses EntityMetadata. You can also reference http://msdn.microsoft.com/en-us/library/microsoft.xrm.sdk.metadata.entitymetadata.aspx for information on the data types and fields available as the JavaScript objects map to the .NET data model pretty closely.

retrieveEntity

Retrieve the metadata for a specific CRM entity. The result is cached for use by subsequent calls.

```
Hsl.Metadata.retrieveEntity(entityName: string, opts?: RetrieveEntityOptions, svcOpts?: OrgService.ServiceOptions):
        Promise<{ EntityMetadata: EntityMetadata; }>;
```

retrieveEntities

Retrieve a set of entities' metadata. The result is cached for use by subsequent calls.

```
Hsl.Metadata.retrieveEntity(entityNames: string[], opts?: RetrieveEntityOptions, svcOpts?: OrgService.ServiceOptions):
        Promise<EntityMetadata[]>;
```

retrieveAllEntities

Retrieves the metadata for all entities in the CRM organization.

```
Hsl.Metadata.retrieveEntity(opts?: RetrieveEntityOptions, svcOpts?: OrgService.ServiceOptions): Promise<{ EntityMetadata:
        EntityMetadata[]; }>;
```

retrieveAttribute

Retrieves the metadata for a specific attribute.

```
Hsl.Metadata.retrieveAttribute(entityName: string, attributeName: string, opts?: RetrieveMetadataOptions, svcOpts?:
        OrgService.ServiceOptions): Promise<{ EntityMetadata: EntityMetadata[]; }>;
```

RetrieveEntityOptions (Interface)

An object defining options for retrieving the entity. It has the following options:

Option	Type	Required	Notes
retrieveAsIfPublished	Boolean	No	Default: true. If true, unpublished metadata is retrieved.
entityFilters	Flag (Integer)	Yes	Default: 1 (Entity only). A flag value specifying the parts of the entity metadata to retrieve. This should be a space separated list of entity filter values or the flags you want to use bitwise-OR'd together. Examples: `{ entityFilters: 1|2/* Entity and Attributes*/ }` `{ entityFilters: "Entity Attributes Relationships Privileges" }` EntityFilters SDK reference For performance and as a best practice, only the parts of the entity metadata that are actually needed should be retrieved.

RetrieveMetadataOptions (interface)

An object defining options for retrieving metadata. It has the following options:

Option	Type	Required	Notes
retrieveAsIfPublished	Boolean	No	Default: true. If true, unpublished metadata is retrieved.