CHEAT SHEET · WEB API

Power Pages Web API

Quick Reference Guide for CRUD Operations & Query Options

Understanding the Power Pages Web API

The **Power Pages Web API** enables client-side JavaScript to perform CRUD operations (Create, Read, Update, Delete) against **Dataverse tables** directly from the browser. Unlike Liquid templates that render server-side, the Web API allows dynamic data interactions after page load, making it essential for building responsive Power Pages applications.

The API follows **OData v4 conventions** and supports powerful query options like `$select`, `$filter`, `$expand`, `$orderby`, and `$top` to retrieve exactly the data you need. All requests must use the `webapi.safeAjax()` wrapper function, which automatically handles CSRF token management required for write operations.

Security note: The Web API respects **Table Permissions** and Web Roles. Users can only access data they are authorized for. Each table must be explicitly enabled for Web API access via site settings, and specific fields can be whitelisted or blacklisted.

Read Operations

GET Basic Read

Retrieve all records from a table using webapi.safeAjax()

```
webapi.safeAjax({
  type: "GET",
  url: "/_api/accounts",
  contentType: "application/json",
  success: function(data) {
    console.log("Records:", data.value);
    data.value.forEach(function(record)
{
      console.log(record.name);
    });
  }
});
```

GET $top - Limit Records

Limit the number of records returned

```
/_api/accounts?$top=10
```

GET $select - Select Specific Fields

Return only specified properties to improve performance

```
/_api/accounts?$select=name,revenue
```

GET $expand - Include Related Records

Retrieve related records via navigation properties

```
/_api/accounts?$expand=contact_customer
```

GET $filter - Filter Results

Apply filtering criteria using OData operators (eq, ne, gt, lt, and, or)

```
/_api/accounts?$filter=revenue gt 90000
```

GET $count - Get Record Count

Return total count of records

```
/_api/accounts?$count=true
```

GET String Functions

Use contains(), startswith(), endswith() for text filtering

```
/_api/accounts?
$filter=contains(name,'Contoso')
```

GET $orderby - Sort Results

Sort results ascending or descending

```
/_api/accounts?$orderby=revenue desc
```

GET Combine Query Options

Chain multiple query options with &

```
/_api/accounts?
$select=name&$filter=revenue gt
1000&$orderby=name&$top=5
```

Write Operations

POST Create Record

Create new record with JSON payload using webapi.safeAjax()

```
webapi.safeAjax({
  type: "POST",
  url: "/_api/accounts",
  contentType: "application/json",
  data: JSON.stringify({
    "name": "Sample Account",
    "revenue": 50000
  }),
  success: function(data) {
    console.log("Record created:",
data.accountid);
  }
});
```

DELETE Delete Record

Delete record by ID using webapi.safeAjax()

```
webapi.safeAjax({
  type: "DELETE",
  url: "/_api/accounts(record-id)",
  success: function() {
    console.log("Record deleted");
  },
  error: function(error) {
    console.error("Delete failed:",
error);
  }
});
```

PATCH Update Record

Update existing record by ID using webapi.safeAjax()

```
webapi.safeAjax({
  type: "PATCH",
  url: "/_api/accounts(record-id)",
  contentType: "application/json",
  data: JSON.stringify({
    "name": "Updated Name",
    "revenue": 100000
  }),
  success: function(data) {
    console.log("Record updated");
  }
});
```

Helper Function

Generic Web API Function

Reusable function that accepts parameters for any Web API operation

```
function callWebAPI(method, entitySet,
recordId, data, queryString) {
  var url = "/_api/" + entitySet;
  if (recordId) url += "(" + recordId +
")";
  if (queryString) url += "?" +
queryString;

  webapi.safeAjax({
    type: method,
    url: url,
    contentType: "application/json",
    data: data ? JSON.stringify(data) :
null,
    success: function(result) {
      console.log("Success:", result);
      return result;
    },
    error: function(error) {
      console.error("Error:", error);
    }
  });
}

// Usage Examples:
callWebAPI("GET", "accounts", null,
null, "$select=name&$top=5");
callWebAPI("POST", "accounts", null,
{name: "New Account"});
callWebAPI("PATCH", "accounts", "record-
id", {name: "Updated"});
callWebAPI("DELETE", "accounts",
"record-id");
```

Dataverse Relationships

GET 1:N - Read Related Records

One-to-Many: Use $expand to retrieve child records (e.g., Account → Contacts)

```
webapi.safeAjax({
  type: "GET",
  url: "/_api/accounts(account-id)?
$expand=contact_customer_accounts",
  success: function(data) {
    console.log("Child contacts:",
data.contact_customer_accounts);
  }
});
```

GET N:1 - Read Parent Record

Many-to-One: Use $expand to retrieve parent record (e.g., Contact → Account)

```
webapi.safeAjax({
  type: "GET",
  url: "/_api/contacts(contact-id)?
$expand=parentcustomerid_account",
  success: function(data) {
    console.log("Parent account:",
data.parentcustomerid_account);
  }
});
```

POST Create with N:1 Lookup

Create child record and link to parent using lookup field binding

```
webapi.safeAjax({
  type: "POST",
  url: "/_api/contacts",
  contentType: "application/json",
  data: JSON.stringify({
    "firstname": "John",
    "lastname": "Doe",

"parentcustomerid_account@odata.bind":
"/accounts(account-id)"
  })
});
```

PATCH Update N:1 Relationship (Method 1)

Update lookup using @odata.bind annotation in PATCH request

```
webapi.safeAjax({
  type: "PATCH",
  url: "/_api/contacts(contact-id)",
  contentType: "application/json",
  data: JSON.stringify({

"parentcustomerid_account@odata.bind":
"/accounts(new-account-id)"
  })
});
```

PUT

Update N:1 Relationship (Method 2)

Change reference using PUT on $ref endpoint with @odata.id

```
webapi.safeAjax({
  type: "PUT",
  url: "/_api/contacts(contact-
id)/parentcustomerid_account/$ref",
  contentType: "application/json",
  data: JSON.stringify({
    "@odata.id": "/_api/accounts(new-
account-id)"
  })
});
```

DELETE

Remove N:1 Lookup (Disassociate)

Remove single-valued navigation property reference using DELETE on $ref endpoint

```
webapi.safeAjax({
  type: "DELETE",
  url: "/_api/contacts(contact-
id)/parentcustomerid_account/$ref",
  success: function() {
    console.log("Lookup cleared");
  }
});
```

GET

N:N - Read Many-to-Many

Many-to-Many: Use $expand with relationship collection (e.g., Contact → Marketing Lists)

```
webapi.safeAjax({
  type: "GET",
  url: "/_api/contacts(contact-id)?
$expand=listmember_association",
  success: function(data) {
    console.log("Marketing lists:",
data.listmember_association);
  }
});
```

POST

N:N - Associate Records

Create Many-to-Many relationship using $ref endpoint

```
webapi.safeAjax({
  type: "POST",
  url: "/_api/contacts(contact-
id)/listmember_association/$ref",
  contentType: "application/json",
  data: JSON.stringify({
    "@odata.id": "/_api/lists(list-id)"
  })
});
```

DELETE N:N - Disassociate Records

Remove Many-to-Many relationship using $ref with $id query parameter

```
webapi.safeAjax({
  type: "DELETE",
  url: "/_api/contacts(contact-
id)/listmember_association/$ref?
$id=/_api/lists(list-id)",
  success: function() {
    console.log("Relationship removed");
  }
});
```

Navigation Property Names

Find relationship names in Power Pages Design Studio or using $metadata endpoint

```
/_api/$metadata
// Look for NavigationProperty elements
// Example: contact_customer_accounts
```

Security & Configuration

Table Permissions Required

Web API respects table permissions. Configure via Portal Management → Table Permissions + Web Roles

```
Users can only access data authorized by their web role
```

CSRF Token Required

Write operations require CSRF token for security. Use webapi.safeAjax()

```
webapi.safeAjax({
  type: "POST",
  url: "/_api/accounts",
  data: JSON.stringify({name: "Test"})
})
```

Enable Web API per Table

Must enable in site settings for each table

```
Webapi/<tablename>/enabled = true
Webapi/<tablename>/fields = name,email,phone
```

Quick Reference Summary

Read Operations (GET)

- → **$select** - Return only specific fields
- → **$filter** - Apply conditions (eq, gt, contains)
- → **$expand** - Include related records
- → **$orderby** - Sort results (asc/desc)
- → **$top** - Limit record count

Write Operations

- → **POST** - Create new records
- → **PATCH** - Update existing records
- → **DELETE** - Remove records
- → **PUT /$ref** - Update relationships

Relationship Patterns

Security Essentials

- → Always use **webapi.safeAjax()**

- **1:N** - $expand on parent for children
- **N:1** - @odata.bind for lookup assignment
- **N:N** - POST/DELETE to /$ref endpoint

- Configure **Table Permissions**
- Enable tables via **Site Settings**
- Whitelist fields explicitly

Pro Tip: Combine query options for optimal performance: `/_api/accounts?$select=name,revenue&$filter=revenue gt 10000&$orderby=name&$top=50`

PowerPortals.de

Need help with Power Pages Web API? Book a consultation at powerportals.de