MRI

API Design Guide

2016

Every released version of MRI products gets its own release notes. However, guides are only updated when changes have been made to product features that require changes to documentation. If there is no guide specific to your version, use the most recent corresponding document (which may refer to an earlier version of the software).

Table of Contents

Chapter 1 Getting Started 5
- Understanding Partner and Developer API Authentication Keys 6

Setting Up an API 6
- Understanding Meta Fields 6
- Creating a New Page in WebDesign 6
- Naming Your API 8

Customizing Your API 8
- Adding a Grid 8
- Adding Columns to Your Grid 9
- Revising the ColumnName Property 10
- Setting Up Input Parameters 11
- Using Input Parameters for Filtering 11

Testing Read-Only APIs 11

Chapter 2 Step-by-Step Scenario Guides 13

Scenario 1: A Filterable API that Reads Data from a Single Table 13
- Building the API 13
- Testing the API 14

Scenario 2: An API that Reads Data from Multiple Tables 14

Scenario 3: An API that Changes Data in a Table 16
- Building the API 16
- Testing the API 16

Scenario 4: An API with Optional Filtering 17
- Testing the API 18

Scenario 5: An API for Saving New Records 20
- Building the API 20
- Testing the API 22

Scenario 6: An API with Custom Paging 23
- Building the API 23
- Testing the API 24

Scenario 7: An API for Nested Relationships 26
- Building the API 26
- Testing the API 28

Scenario 8: An API for Validating Input Parameters 28
- Building the API 29
- Testing the API 29

Scenario 9: Improving Performance of an API thatChanges Data in a Table 32
- Building the API 33
- Testing the API 33

Scenario 10: An API That Uses Query String Values While Changing the Data 34
- Building the API 34
- Testing the API 35

Appendix A Product Codes 37

Appendix B Shaping Responses 38

Getting Started 1

This guide will cover how to set up and test an API using APIDesign. You should be familiar with building pages using WebDesign. You may need to have the *WebDesign User Guide* available for reference.

Before you begin, you need to create a Web Services role that will be dedicated to testing your APIs. You will assign this role to a Web Services user to control which APIs can be executed. For more information on adding a Web Services role, creating a Web Services user, and assigning Web Services roles to Web Services users, refer to the *System Administration Guide*.

Understanding Partner and Developer API Authentication Keys

MRI will provide an authentication key in one of two forms. Partner keys only execute APIs that are in the manifest. They have a call rate of 1,000 requests per five-minute rolling window, and their API structure is cached for 24 hours. Developer keys allow non-registered APIs to be executed. They have a lower call rate than partner keys, and their API structure is newly loaded every time and never cached.

Setting Up an API

You must create a new activity group that will contain all APIs that you create. You should create this activity group before you begin. For more information on creating new activity groups, refer to the *WebDesign User Guide*.

To set up a new API, you will need to complete the following steps:

- Create a new page, as described in "Creating a New Page in WebDesign" on page 6.
- Name your API, as described in "Naming Your API" on page 8.

Understanding Meta Fields

When creating an API, you will use something called a Meta field to specify certain behaviors of the API. A Meta field is a calculation field with certain properties defined in a particular way.

To add a Meta field to your page, follow these steps:

1. Add a calculation field to the page, as described in the *WebDesign User Guide*.
2. Right-click the field, click **Display Properties**, and then click the **Properties** tab.
3. In the **VariableName** property, enter **META.X**, where **X** is the name of the Meta field that you are adding.
4. To specify a fixed value for the Meta field, enter that value in the **Default** property.

Note

For the various instances where you need to add Meta fields, the values for the **VariableName** and **Default** properties will be provided to you.

Creating a New Page in WebDesign

To create a new page in WebDesign for your API, follow these steps:

1. Click **Application Toolkit**, and then click **WebDesign** to display the **WebDesign** view and the **Open Page/Group** dialog box.

   Note

   If you already have WebDesign open, click the **Open Activity Group** button to open the **Open Page/Group** dialog box.

2. Click the activity group folder to which you want to add a new page, and then click **Open** to open the first page of the selected activity group in the **WebDesign** view. You can also click a page in the right frame, and then click **Open** to open it in the **WebDesign** view.

3. Click the **New Page** button, and then complete the following fields to define the new page for your API:

   - **Page ID**—Enter a unique name for the page.
   - **Description**—Enter a page description to appear in the title bar for the page and in views and dialog boxes in WebDesign.
   - **Table**—Do not specify a table.
   - **Data from Tables Will Only Be Displayed in a Grid**—Do not select this check box.

4. Name the API, as described in "Naming Your API" on page 8.

Naming Your API

The name of your API will be part of the URL path used to invoke the API.

Example

If you name your API **AP_Vendors**, your API will be accessible at the following URL:
https://[MRIWebDomain]/mriweb/mriapiservices/api.asp?$api=**AP_Vendors**

To name your API, add a Meta field to the page, as described in "Understanding Meta Fields" on page 6.

- In the **VariableName** property, enter **META.APINAME**. This defines your page as an API.
- In the **Default** property, enter the name of your API. Name your API using the following format:

  C[your Salesforce account auto-number]_[product code]_[descriptive name]

  You can request your Salesforce account auto-number from your MRI account executive. For a list of product codes, refer to "Product Codes" on page 37.

  Caution!
  When submitting an API for inclusion in the manifest, it may be rejected during the approval process if it does not have the proper naming format.

Customizing Your API

This section contains instructions for various components that you can use to customize your API. For examples of how and when to use these components, refer to "Step-by-Step Scenario Guides" on page 13.

Adding a Grid

In order for an API to be useful, it needs to have a fixed shape. APIDesign uses grids to define this shape.

To add a grid to your page, follow these steps:

1. Add a grid to the page, as described in the *WebDesign User Guide*.
2. If you want to expose data directly from a table through your API, when the **Select Grid's Table** dialog box appears, select the table that contains the information that you want to expose through the API.
3. Right-click the grid, click **Display Properties**, and then click the **Properties** tab.

4. In the **GridId** property, ensure that the value is a short descriptive name with no spaces.

   Note

   If you linked your grid to a table when you first added the grid, the **GridId** property will be set to the name of that table.

5. Proceed to "Adding Columns to Your Grid" on page 9 to organize how the data in your API is displayed in the API response.

Adding Columns to Your Grid

By adding columns to your grid, you can expose certain properties of each record output by the API. To add columns to your grid, follow these steps:

1. Click **Display Field Information**  .
2. On the **Table Fields** dialog box (Figure 1-1), in the **Selected Object** field, select
3.
4.
5. your grid to display the table fields that are related to the table on which your grid is based.

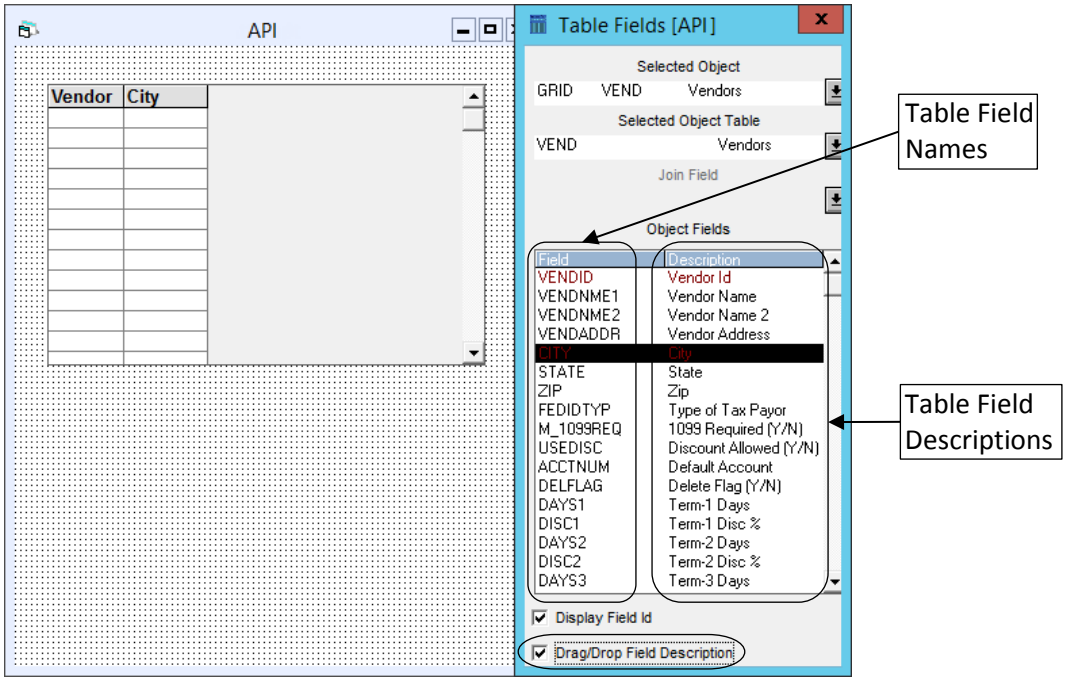


Figure 1-1. Table Fields Dialog Box

6. Select the **Drag/Drop Field Description** check box (Figure 1-1). Selecting this check box will set the **ColumnName** property of each column that you add to

your grid to the corresponding table field description in the **Description** column of the **Object Fields** area.

7. In the **Object Fields** area, select the table fields that you want to add as columns in the grid, and then drag each table field to a space within the grid.
8. Review the **ColumnName** property for each column in your grid and revise as necessary. To review and revise the **ColumnName** property, proceed to "Revising the ColumnName Property" on page 10.

Revising the ColumnName Property

The **ColumnName** property determines the name of the data nodes that are returned when you call your API. You should review the **ColumnName** property for each of your columns to ensure that the names are explicit and do not contain swapnames.

Caution!
If a **ColumnName** property contains a swapname, the API will fail. Replace the swapname with a static name to ensure that API callers are not affected by changes to your swapnames.

To replace the value in the **ColumnName** property, follow these steps:

1. Right-click the column, click **Display Properties**, and then click the **Column** tab.
2. If necessary, replace the value in the **ColumnName** property with a static, explicit name.

Note

Any spaces in the **ColumnName** property are ignored when the API is processed.

Example

The **ColumnName** property for the VENDID table field is set to **!{Vendor} Id** by default. Replace the value in the **ColumnName** property with **VendorID**.

Setting Up Input Parameters

Filtering is controlled by input parameters. Callers will provide values for input parameters when they call the API. You should always specify filtering to control the size of the data that will be exposed to the API.

To add an input parameter, follow these steps:

1. Add a calculation field to the page, as described in the *WebDesign User Guide*.
2. Right-click the calculation field, click **Display Properties**, and then click the **Properties** tab.
3. In the **VariableName** property, enter a descriptive name.

Example

If you want to filter by the RMPROPID table field, you might set the **VariableName** property of the calculation field to **PROPERTY**.
If you want to filter by the CITY table field, you might set the **VariableName** property of the calculation field to **CITY**.

Using Input Parameters for Filtering

When you are finished adding input parameters, you can use them in expressions on the grid to control how data is loaded. To add an expression to the grid to link the input parameters to their respective columns, follow these steps:

1. On the **Grid Properties** dialog box, click the **Properties** tab, and then double-click the **Query** property.
2. In the **Expressions** dialog box, enter the following expression to specify how to filter the grid using the input parameters:

   table field name=[*VariableName property value*]

Note

If you have multiple input parameters that you want to use as filters, separate the expressions with **AND**.

Example

If the table field is RMPROPID, and you set the **VariableName** property to **PROPERTY**, you would enter the following for your expression:
RMPROPID=[PROPERTY]

Testing Read-Only APIs

You can test read-only APIs in any browser. To test your API, follow these steps:

1. Give your Web Services role access to your API, as described in the *System Administration Guide*.
2. Open any browser and enter the following URL for the API:

   Root URL for where MRI APIs are installed. This is specific to your environment and will be provided to you by your system administrator.

   http://[MRIWebDomain]/mriapiservices/api.asp?$api=[your API name]

3. When prompted, enter your credentials.
   - **User Name**—Your user name is composed of the client ID, database name, user name, and partner key, separated by forward slashes.
   - **Password**—Your password is the password for your Web Services user name.

   Example

   Your client ID is **C1**, your MRI application database name is **D1**, your Web Services user name is **U1**, your assigned MIX partner key is **P1**, and your Web Services user password is **Password**.
   With the above information, you would enter **C1/D1/U1/P1** in the **User Name** field and **Password** in the **Password** field.

4. Review the results. You may want to experiment with different shapes of data by varying the $format query string parameter. For a full listing of available formats, refer to "Shaping Responses" on page 38.

   Example

   To see the data in plain XML, append **$format=xml** to the query string:
   http://[MRIWebDomain]/mriapiservices/api.asp?$api=[your API name]&**$format=xml**
   Refresh the page to see the data in the new format.

Step-by-Step Scenario Guides 2

The following scenarios describe different ways to achieve the desired shape and behavior of your APIs.

You can follow along with these scenarios to create the example API, and then repeat the steps for your specific case

Scenario 1: A Filterable API that Reads Data from a Single Table

In this scenario, you will create a read-only API named Vendors. This API will read from the Vendor table (VEND) and filter by the CITY and STATE table fields.

Building the API

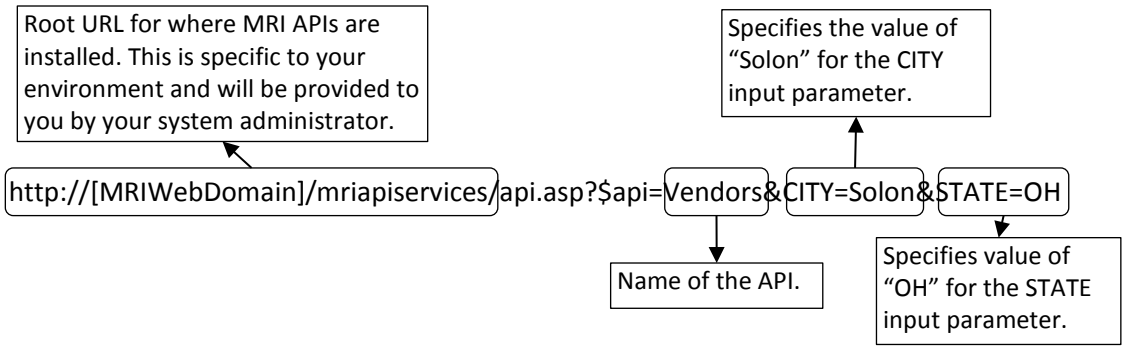
To build this API, follow these steps:

1. Follow the instructions to set up your API in "Setting Up an API" on page 6.
2. Add a grid to the page, as described in "Adding a Grid" on page 8. When prompted to select a table, select the VEND table. Leave the **GridId** property set to **VEND**.
3. Add columns to the grid, as described in "Adding Columns to Your Grid" on page 9. For this scenario, drag and drop the VENDID, VENDNME1, CITY, STATE, ZIP, STATUS, and CONTACT table fields into the grid on the page.
4. Review the **ColumnName** properties for each column and revise as necessary, as described in "Revising the ColumnName Property" on page 10. For this scenario, make the following replacements:
   - For the VENDID table field, in the **ColumnName** property, replace **!{Vendor} Id** with **VendorID**.
   - For the VENDNME1 table field, in the **ColumnName** property, replace **!{Vendor} Name** with **VendorName**.
5. In the **Grid Properties** dialog box, on the **Properties** tab, set the **ViewOnly** property to **Yes**. Setting this property to **Yes** makes the data read-only and enables automatic paging of data. For more information on paging, refer to the *API Integrations Guide*.
6. Add input parameters, as described in "Setting Up Input Parameters" on page 11. For this scenario, add two input parameters. Set the **VariableName** property to **CITY** for the first input parameter and to **STATE** for the second input parameter.
7. Add an expression to the grid for filtering purposes, as described in "Using Input Parameters for Filtering" on page 11. For this scenario, enter **CITY=[CITY] AND STATE=[STATE]** in the **Expressions** dialog box.

Testing the API

To test the Vendors API in any browser, follow these steps:

1. Give your Web Services role access to your API, as described in the *System Administration Guide*. For this scenario, in the **Access** column, select the check box next to the Vendors API.
2. Open any browser and go to the following URL for the Vendors API:



3. Enter your credentials, as described in "Testing Read-Only APIs" on page 11.
4. Review the results.

Scenario 2: An API that Reads Data from Multiple Tables

In this scenario, you will create a read-only API named Vendors. This API will read from the Vendor table (VEND) and also include the **DESCRPTN** column from the related Vendor Payment Priority table (VENDPRIORITY).

Building the API

To build this API, follow these steps:

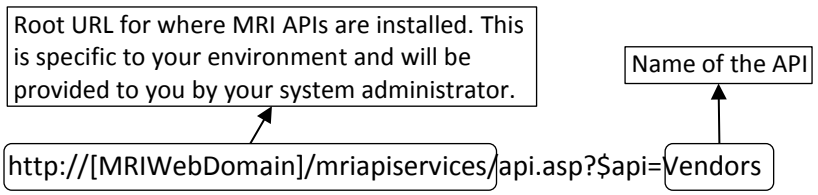
1. Follow the instructions to set up your API in "Setting Up an API" on page **Error! Bookmark not defined.**.
2. Add a grid to the page, as described in "Adding a Grid" on page 8. When prompted to select a table, select the VEND table. Leave the **GridId** property set to **VEND**.
3. Add columns to the grid, as described in "Adding Columns to Your Grid" on page 9. For this scenario, drag and drop the VENDID, VENDNME1, CITY, STATE, ZIP, STATUS, and CONTACT table fields into the grid on the page.

4. Add a column from a related table by following these steps:
   a. On the **Table Fields** dialog box, in the **Selected Object Table** field, select the Vendor Payment Priority table (VENDPRIORITY).
   b. Drag and drop the DESCRPN table field into the grid on the page.
5. Review the **ColumnName** properties for each column and revise as necessary, as described in "Revising the ColumnName Property" on page 10. For this scenario, make the following replacements:
   - For the VENDID table field, in the **ColumnName** property, replace **!{Vendor} Id** with **VendorID**.
   - For the VENDNME1 table field, in the **ColumnName** property, replace **!{Vendor} Name** with **VendorName**.
6. On the **Properties** tab in the **Grid Properties** dialog box, set the **ViewOnly** property to **Yes**. Setting this property to **Yes** makes the data read-only and enables automatic paging of data. For more information on paging, refer to the *API Integrations Guide*.

Testing the API

To test the Vendors API in any browser, follow these steps:

1. Give your Web Services role access to your API, as described in the *System Administration Guide*. For this scenario, in the **Access** column, select the check box next to the Vendors API.
2. Open any browser and go to the following URL for the Vendors API:



3. Enter your credentials, as described in "Testing Read-Only APIs" on page 11.
4. Review the results.

Scenario 3: An API that Changes Data in a Table

In this scenario, you will enhance the API created in "Scenario 1: A Filterable API that Reads Data from a Single Table" on page 13 to be able to insert or modify data in the VEND table.

Building the API

To add to the Vendors API from Scenario 1, follow these steps:

1. Open the Vendors API.
2. Add a Meta field, as described in "Understanding Meta Fields" on page 6.
   a. In the **VariableName** property, enter **META.ID.[GridId]**. For this scenario, enter **META.ID.VEND**.
   b. In the **Default** property, enter the table field name to use as the primary key for this grid. For this scenario, enter **VENDID**.
3. Add a command button, as described in the *WebDesign User Guide*.
4. Right-click the new command button, and then click **Display Properties** to open the **Command Button Properties** dialog box.
5. On the **Appearance** tab, in the **Text** property, enter **Save**. Entering **Save** in the **Text** property allows the API to save changes to the data defined by the grid.

Testing the API

This API cannot be tested in a browser. MRI Software recommends using an HTTP debugging proxy server software. If your company does not allow you to install proxy software, you can search for "HTTP" in your browser's online store to find a browser application or extension to help you issue HTTP requests to your API. To test this API, follow these steps:

1. Update the following request with your information, and then use your HTTP tool to issue the request:

   Note

   For more information on how to invoke APIs that change data, refer to the *API Integrations Guide*.

```
POST http://[MRIWebDomain]/mriapiservices/api.asp?$api=[API
     name] HTTP/1.1
Authorization: Basic QzEvRDEvVTEvUDE6UGFzc3dvcmQ=
Content-Type: application/xml
Accept: application/xml
Content-Length: [depends on size below]

<?xml version="1.0" encoding="utf-8"?>
```

```
<Vendors>
  <entry>
    <VendorID>VEND1</VendorID>
    <VendorName>Sample Vendor Name 1</VendorName>
    <City>Cleveland</City>
    <State>Ohio</State>
    <Zip>44444</Zip>
    <Status>A</Status>
    <Contact>Sample Name 1</Contact>
  </entry>
  < entry>
    <VendorID>VEND2</VendorID>
    <VendorName>Sample Vendor Name 2</VendorName>
    <City>Cleveland</City>
    <State>Ohio</State>
    <Zip>44444</Zip>
    <Status>A</Status>
    <Contact>Sample Name 2</Contact>
  </entry>
</Vendors>
```

2. Verify that you received a response with HTTP status code of 200. You should see any new vendors in MRI. For this scenario, you should see Sample Vendor Name 1 and Sample Vendor Name 2. For more information on creating or modifying data with APIs, refer to the *API Integrations Guide*.

Scenario 4: An API with Optional Filtering

In this scenario, you will create a read-only API named Units. This API will read from the Units table (UNIT) and filter by the RMPROPID and RMBLDGID table fields, and optionally by the UNITID table field.

Building the API

To build this API, follow these steps:

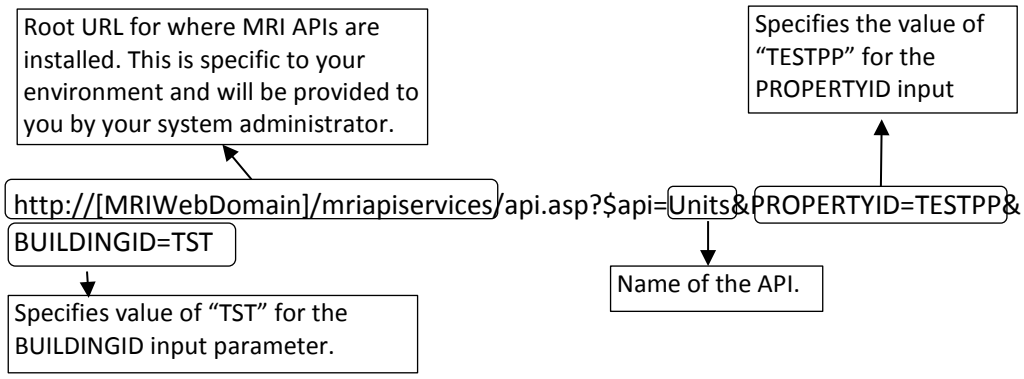
1. Follow the instructions to set up your API in “Setting Up an API” on page 6
2. Add a grid to the page, as described in “Adding a Grid” on page 8. When prompted to select a table, select the UNIT table. Leave the **GridId** property set to **UNIT**.
3. Add columns to the grid, as described in “Adding Columns to Your Grid” on page 9. For this scenario, drag and drop the RMPROPID, RMBLDGID, UNITID, ADDRESS, CLASSID, SQFT, MOVEIN, MOVEOUT, and BASERENT table fields into the grid on the page.

4. Review the **ColumnName** properties for each column and revise as necessary, as described in "Revising the ColumnName Property" on page 10. For this scenario, make the following replacements:
   - For the RMPROPID table field, in the **ColumnName** property, replace **!{RM Property} Id** with **PropertyID**.
   - For the RMBLDGID table field, in the **ColumnName** property, replace **!{RM Building} Id** with **BuildingID**.
   - For the UNITID table field, in the **ColumnName** property, replace **!{Unit} Id** with **UnitID**.
   - For the SQFT table field, in the **ColumnName** property, replace **!{Square Foot}** with **SquareFootage**.
5. In the **Grid Properties** dialog box, on the **Properties** tab, set the **ViewOnly** property to **Yes**. Setting this property to **Yes** makes the data read-only and enables automatic paging of data. For more information on paging, refer to the *API Integrations Guide*.
6. Add input parameters, as described in "Setting Up Input Parameters" on page 11. For this scenario, add three input parameters. Set the **VariableName** property to **PROPERTYID** for the first input parameter, **BUILDINGID** for the second input parameter, and **UNITID** for the third input parameter.
7. Add an expression to the grid for filtering purposes, as described in "Using Input Parameters for Filtering" on page 11. For this scenario, enter **RMPROPID=[PROPERTYID] AND RMBLDGID=[BUILDINGID] IF{[UNITID]=[*],,AND UNITID=[UNITID]}** in the **Expressions** dialog box. The last part of this expression is a conditional statement that first tests to see if the **UNITID** parameter is blank, and if it is not, it appends **AND UNITID=[UNITID]** to the end of the filter query.

Testing the API

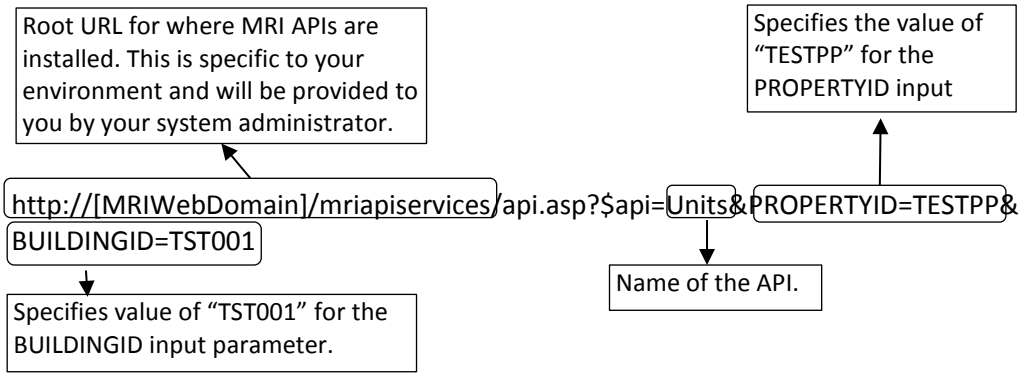
To test the Units API in any browser, follow these steps:

1. Give your Web Services role access to your API, as described in the *System Administration Guide*. For this scenario, in the **Access** column, select the check box next to the Units API.
2. To test leaving the optional filter blank, open any browser and go to the following URL for the Units API:



3. Enter your credentials, as described in "Testing Read-Only APIs" on page 11.
4. Review the results. You should see all units for the property and building specified in the query string.

5. To test functionality of the optional filter, open any browser and go to the following URL for the Units API:



6. Enter your credentials, as described in "Testing Read-Only APIs" on page 11.
7. Review the results. You should see all units for the property and building specified in the query string.

Scenario 5: An API for Saving New Records

In this scenario, you will create a write-only API called Residents. This API will allow the caller to create a new record in the Resident Names table (NAME).

Building the API

To build this API, follow these steps:

1. Follow the instructions to set up your API in "Setting Up an API" on page 6.
2. Add a grid to the page, as described in "Adding a Grid" on page 8. When prompted to select a table, select the NAME table. Leave the **GridId** property set to **NAME**.
3. Add columns to the grid, as described in "Adding Columns to Your Grid" on page 9. For this scenario, drag and drop the NAMEID, PHONE, RESPROPID, RMBLDGID, UNITID, RMLEASE, EMAIL, and CELL table fields into the grid on the page.
4. Right-click the grid, click **Display Properties**, and then click the **Column** tab.
5. Add a calculation column by clicking **Add Calc Col**, and then set the **ColumnName** property to **SiteID** and the **VariableName** property to **SITEID**.
6. Review the **ColumnName** properties for each column and revise as necessary, as described in "Revising the ColumnName Property" on page 10. For this scenario, make the following replacements:
   - For the NAMEID table field, in the **ColumnName** property, replace **!{RM Name} Id** with **ResidentID**.

- For the RESPROPID table field, in the **ColumnName** property, replace **!{RM Property} Id** with **PropertyID.**
- For the RMBLDGID table field, in the **ColumnName** property, replace **!{RM Building} Id** with **BuildingID**.
- For the UNITID table field, in the **ColumnName** property, replace **!{RM Unit} Id** with **UnitID**.
- For the RMLEASE table field, in the **ColumnName** property, replace **!{RM Lease} Id** with **LeaseID**.

7. Add a Meta field, as described in "Understanding Meta Fields" on page 6.
   a. In the **VariableName** property, enter **META.ID.[GridId]**. For this scenario, enter **META.ID.NAME**.
   b. In the **Default** property, enter the table field name to use as the primary key for this grid. For this scenario, enter **NAME.NAMEID**.

Note

This Meta field identifies the primary keys of the table to which data is being saved.

8. Add another Meta field, as described in "Understanding Meta Fields" on page 6.
   a. In the **VariableName** property, enter **META.ALLOW**.
   b. In the **Default** property, enter **POST**.

Note

This Meta field identifies all valid methods of calling this API. Because this is a write-only API, only the POST verb is allowed.

9. Add an expression to the grid for filtering purposes, as described in "Using Input Parameters for Filtering" on page 11. For this scenario, enter **1=0** in the **Expressions** dialog box. This prevents the grid from loading any records from the NAME table, because they are not necessary for creating a new record.
10. In the **PreInsertTrigger** property for the **NAMEID** column, enter the following:

```
IF{SELECT{TOP 1 SITEID FROM SITEMAP (NOLOCK) WHERE ENTITYID
IN(SELECT ENTITYID FROM RMPROP (NOLOCK) WHERE
RMPROPID='CELL{NAME.RESPROPID}') AND SITEID<>'HO' AND
SITEID<>'@', [SITEID]},
IF{BLANK{SITEID} OR LENGTH{CELL{SITEID}}=0,
SETFLD{NAME.NAMEID,@VAUTONUM{NAMEID,,9}}SETFLD{SITEID,@},
SETFLD{NAME.NAMEID,CELL{SITEID}VAUTONUM{NAMEID,,8}}},
IF{BLANK{SITEID} OR LENGTH{CELL{SITEID}}=0,
SETFLD{NAME.NAMEID,@VAUTONUM{NAMEID,,9}}SETFLD{SITEID,@},
SETFLD{NAME.NAMEID,CELL{SITEID}VAUTONUM{NAMEID,,8}}}}
```

This code tells the API to first retrieve the site ID to which the RM property is tied, and then create a new resident ID using the VAUTONUM function. The NAMEID table field has a length of 10 characters, so if a site ID was found for the property, then the 2-character site ID is prepended and the resident ID is created using 8

characters. If no site ID was found, the default site ID of **@** is prepended and the resident ID is created using 9 characters.

11. Add a command button, as described in the *WebDesign User Guide*.
12. Right-click the new command button, and then click **Display Properties** to open the **Command Button Properties** dialog box.
13. On the **Appearance** tab, in the **Text** property, enter **Save**. Entering **Save** in the **Text** property allows the API to save changes to the data defined by the grid.

Testing the API

This API cannot be tested in a browser. MRI Software recommends using an HTTP debugging proxy server software. If your company does not allow you to install proxy software, you can search for "HTTP" in your browser's online store to find a browser application or extension to help you issue HTTP requests to your API. To test this API, follow these steps:

1. Update the following request with your information, and then use your HTTP tool to issue the request:

   Note

   For more information on how to invoke APIs that change data, refer to the *API Integrations Guide*.

```
POST http://[MRIWebDomain]/mriapiservices/api.asp?$api=[API
     name] HTTP/1.1
Authorization: Basic QzEvRDEvVTEvUDE6UGFzc3dvcmQ=
Content-Type: application/xml
Accept: application/xml
Content-Length: [depends on size below]

<?xml version="1.0" encoding="utf-8"?>
<Residents>
  <entry>
    <PropertyID>Any existing PropertyID</PropertyID>
    <BuildingID>Any existing BuildingID tied to that
       property</BuildingID>
    <UnitID>Any existing UnitID tied to that
       building</UnitID>
    <LeaseID>Any existing LeaseID tied to that property and
       building and unit</LeaseID>
    <EMailAddress>test@test.test</EMailAddress>
    <PhoneNumber>1112223333</PhoneNumber>
    <Cell>4445556666</Cell>
  </entry>
</Residents>
```

2. Verify that you received a response with HTTP status code of 200. You should see any new residents in MRI. For this scenario, you should see your new resident by searching for the resident in MRI. For more information on creating or modifying data with APIs, refer to the *API Integrations Guide*.

Scenario 6: An API with Custom Paging

In this scenario, you will create a read-only API named Units. This API will read from the Units table (UNIT) and filter by the RMPROPID table field.

Building the API

To build this API, follow these steps:

1. Follow the instructions to set up your API in "Setting Up an API" on page 6.
2. Add a grid to the page, as described in "Adding a Grid" on page 8. When prompted to select a table, select the UNIT table. Leave the **GridId** property set to **UNIT**.
3. Right-click the grid, click **Display Properties**, and then click the **Column** tab.
4. Add nine calculation columns by clicking **Add Calc Col** nine times.
5. Modify the **ColumnName** properties for each calculation column, as described in "Revising the ColumnName Property" on page 10. For this scenario, enter the following column names, in order: RMPROPID, RMBLDGID, UNITID, ADDRESS, CLASSID, SQFT, MOVEIN, MOVEOUT, and BASERENT.
6. In the **Grid Properties** dialog box, on the **Properties** tab, ensure that the **ViewOnly** poperty is set**No**. Setting this property to **No** disables automatic paging, allowing custom paging to be implemented.
7. Add input parameters, as described in "Setting Up Input Parameters" on page 11. For this scenario, add three input parameters with the following settings:
   - For the first input parameter, set the **VariableName** property to **PROPERTYID**.
   - For the second input parameter, set the **VariableName** property to **META.TOP**, and the **Default** property to **300**.

     Note

     This setting designates that 300 entries will be returned per API call. You can set the **Default** property to return up to 1,000 entries per API call.

   - For the third input parameter, set the **VariableName** property to **META.SKIP**, and the **Default** property to **0**.

     Note

     This setting designates that the data will be returned starting from the first record in the data setand the framework will automatically increment this value as the caller retrieves subsequent pages.
8. Create a user-defined function in SQL. For more information on how to add a user-defined function, refer to the "Creating and Modifying User-Defined Functions" section of the *Database Design User Guide*.

Example

For this scenario, the user-defined function in Database Design that implements custom paging for an API should be similar to the following:

```
CREATE FUNCTION [dbo].[udf_GetResidentialUnits]
(
  @skip bigint,
  @top int,
  @propertyID varchar(15)
)
RETURNS TABLE
AS
RETURN
(
SELECT
  PROPERTYID, BUILDINGID, UNITID, ADDRESS, CLASSID,
SQUAREFOOTAGE, MOVEIN, MOVEOUT,
  BASERENT, ROW_NUMBER
FROM (
  SELECT
    PROPERTYID, BUILDINGID, UNITID, ADDRESS, CLASSID,
SQUAREFOOTAGE, MOVEIN, MOVEOUT,
    BASERENT, ROW_NUMBER() OVER (ORDER BY PROPERTYID,
BUILDINGID, UNITID) AS ROW_NUMBER
  FROM(
    SELECT
      UNIT.RMPROPID AS PROPERTYID,
      UNIT.RMBLDGID AS BUILDINGID,
      UNIT.UNITID,
      UNIT.ADDRESS,
      UNIT.CLASSID,
      UNIT.SQFT AS SQUAREFOOTAGE,
      UNIT.MOVEIN,
      UNIT.MOVEOUT,
      UNIT.BASERENT
    FROM UNIT
    WHERE UNIT.RMPROPID=@propertyID
    ) AS UNITS
  ) NUMBERED
  WHERE NUMBERED.ROW_NUMBER BETWEEN @skip AND (@skip + @top)
```

9. Enter the following into the **Select** property of the main grid.:

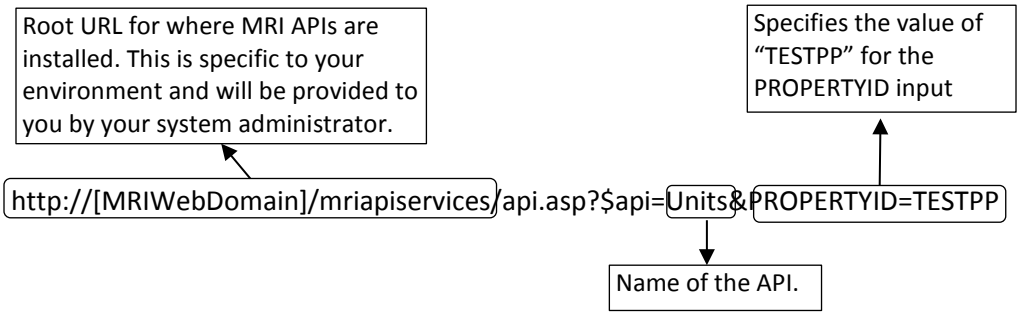
```
SELECT{* from dbo.udf_GetResidentialUnits([META.SKIP],
[META.TOP], '[PROPERTYID]') , [UNIT.RMPROPID] [UNIT.RMBLDGID]
[UNIT.UNITID] [UNIT.SQFT] [UNIT.ADDRESS] [UNIT.CLASSID]
[UNIT.SQFT] [UNIT.MOVEIN] [UNIT.MOVEOUT] [UNIT.BASERENT]}
```

Testing the API

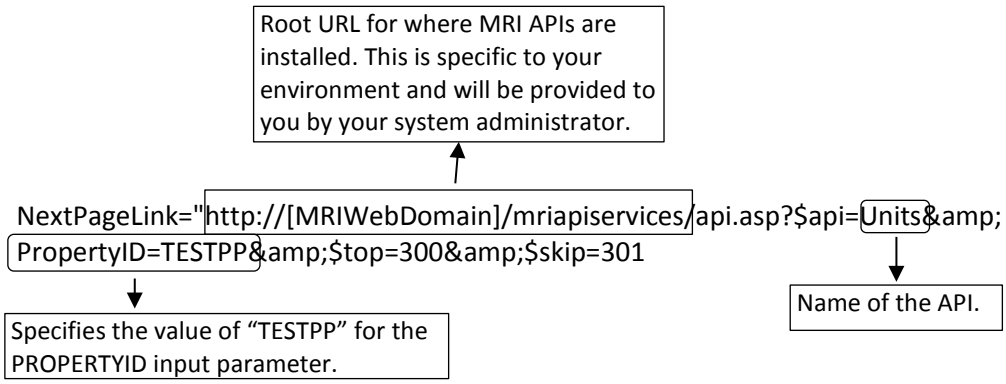
To test the Units API in any browser, follow these steps:

1. Give your Web Services role access to your API, as described in the *System Administration Guide*. For this scenario, in the **Access** column, select the check box next to the Units API.

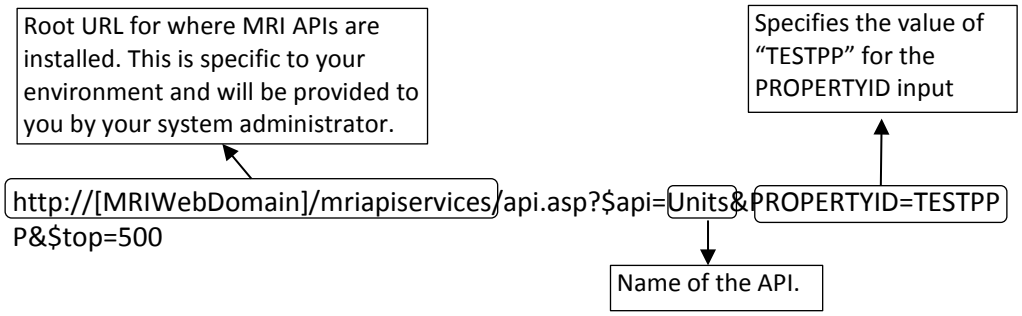
2. To test the default values for the paging filters, open any browser and go to the following URL for the Units API:



3. Enter your credentials, as described in "Testing Read-Only APIs" on page 11.
4. Review the results. You should see the first 300 units for the property specified in the query string .
5. To test the functionality of the custom paging, go to the URL within the root XML node that looks similar to the following:



6. Enter your credentials, as described in "Testing Read-Only APIs" on page 11.
7. Review the results. You should see the second set of 300 units for the property specified in the query string.
8. Repeat Steps 5 through 7 until you have viewed all of the entries. The URL will be returned on all subsequent pages until all of the entries have been shown.
9. To test the functionality of different page sizes, open any browser and go to the following URL for the Units API:



10.

11. Enter your credentials, as described in "Testing Read-Only APIs" on page 11.

12. Review the results. You should now see the first 500 units for for the property specified in the query string.

Scenario 7: An API for Nested Relationships

In this scenario, you will create a read-only API named BuildingUnits. This API will read from the RM Buildings table (RMBLDG) and the Units table (UNIT).

Building the API

To build this API, follow these steps:

1. Follow the instructions to set up your API in "Setting Up an API" on page 6.

2. Add a grid to the page, as described in "Adding a Grid" on page 8. When prompted to select a table, select the RMBLDG table. Leave the **GridId** property set to **RMBLDG**.

3. Add columns to the grid, as described in "Adding Columns to Your Grid" on page 9. For this scenario, drag and drop the RMPROPID, RMBLDGID, NMBRUNIT, DESCRPTN, ADDRESS, CITY, STATE, ZIP, and COUNTY table fields into the grid on the page.

4. Review the **ColumnName** properties for each column and revise as necessary, as described in "Revising the ColumnName Property" on page 10.

   - For the RMPROPID table field, in the **ColumnName** property, replace **!{RM Name} Id** with **PropertyID**.
   - For the RMBLDGID table field, in the **ColumnName** property, replace **!{RM Name} Id** with **BuildingID**.
   - For the NMBRUNIT table field, in the **ColumnName** property, replace **!{RM Name} Id** with **Unit**.

5.

6. In the **Grid Properties** dialog box of the **RMBLDG** grid, on the **Properties** tab, set the **ViewOnly** property to **No**. For more information on paging, refer to the *API Integrations Guide*.
7. Add another grid to the page, as described in "Adding a Grid" on page 8. When prompted to select a table, select the UNIT table. Leave the **GridId** property set to **UNIT**.
8. Add columns to the **UNIT** grid, as described in "Adding Columns to Your Grid" on page 9. For this scenario, drag and drop the RMPROPID, RMBLDGID, UNITID, ADDRESS, CLASSID, SQFT, MOVEIN, MOVEOUT, and BASERENT table fields into the grid on the page.
9. Review the **ColumnName** properties for each column and revise as necessary, as described in "Revising the ColumnName Property" on page 10.
   - For the RMPROPID table field, in the **ColumnName** property, replace **!{RM Name} Id** with **PropertyID**.
   - For the RMBLDGID table field, in the **ColumnName** property, replace **!{RM Name} Id** with **BuildingID**.
   - For the NMBRUNIT table field, in the **ColumnName** property, replace **!{RM Name} Id** with **Unit**.
   - For the SQFT table field, in the **ColumnName** property, replace **!{Square Foot}** with **SquareFootage**
10. Add a calculation column to the topmost grid by clicking **Add Calc Col** on the **Grid Properties** dialog box, and then change the following properties:
    - **ColumnName**—Set this property to **Units**.
    - **VariableName**—Set this property to **META.JOIN.UNIT**. This follows the META.JOIN.X naming scheme, where **X** is the value of the **GridId** property of the sub-grid.

      Note

      If the table name of the table that the sub-grid is tied to contains any underscores, you must remove the underscores from the **GridId** property.

    - **Default**—Set this property to **RMBLDG.RMPROPID=UNIT.RMPROPID AND RMBLDG.RMBLDGID=UNIT.RMBLDGID**. This allows for proper nesting of units to their respective buildings.
11. In the **Grid Properties** dialog box of the **UNIT** grid, on the **Properties** tab, verify that the **ViewOnly** property is set to **No**.
12. Add input parameters, as described in "Setting Up Input Parameters" on page 11. For this scenario, add one input parameter, and set the **VariableName** property to **PROPERTYID**.

13. Add an expression to the **RMBLDG** grid for filtering purposes, as described in "Using Input Parameters for Filtering" on page 11. For this scenario, enter **RMPROPID=[PROPERTYID]** in the **Expressions** dialog box.

14. Add an expression to the **UNIT** grid for filtering purposes, as described in "Using Input Parameters for Filtering" on page 11. For this scenario, enter **UNIT.RMPROPID=[PROPERTYID]** in the **Expressions** dialog box.

Note

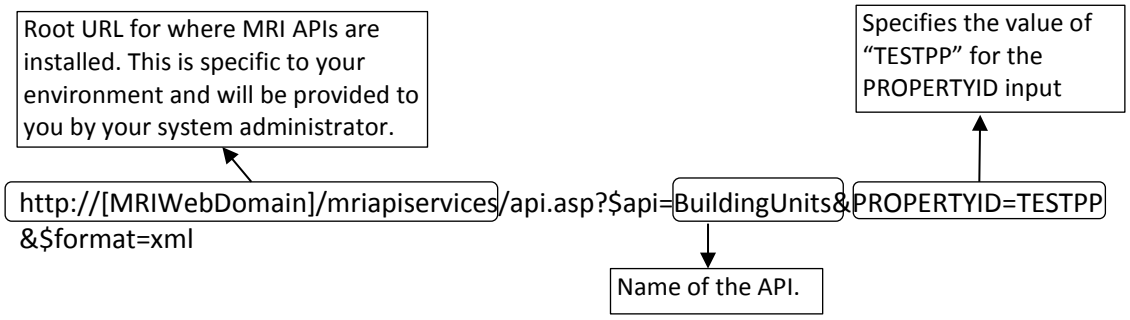
To improve performance, filtering expressions should always be used on **all** grids in an API.

Testing the API

To test the BuildingUnits API in any browser, follow these steps:

1. Give your Web Services role access to your API, as described in the *System Administration Guide*. For this scenario, in the **Access** column, select the check box next to the BuildingUnits API.

2. Open any browser and go to the following URL for the BuildingUnits API:



3. Enter your credentials, as described in "Testing Read-Only APIs" on page 11.

4. Review the results. You should see all buildings for the property specified in the query string, and all of the units that belong to a building listed below the appropriate building.

Scenario 8: An API for Validating Input Parameters

In this scenario, you will enhance the Units API created in "Scenario 4: An API for Optional Filtering" on page 17 to ensure that the caller has to provide a property ID and building ID and the request will fail if they are not both provided.

Building the API

To add to the Units API from Scenario 4, follow these steps:

1. Open the Units API.
2. Right-click the grid, and then click **Display Properties** to open the **Grid Properties** dialog box.
3. On the **Properties** tab, double-click the **PreLoadTrigger** property.
4. In the **Expressions** dialog box, enter the following expression to run all validation checks that are on the page:

   VALIDPAGE{}

5. Right-click the **PROPERTYID** input parameter, and then click **Display Properties** to open the **Calculation Field Properties** dialog box.
6. On the **Properties** tab, double-click the **Validation** property.
7. In the **Expressions** dialog box, enter the following expression:

   VALIDATE{IF{[PROPERTYID]=[*] AND '[PROPERTYID]'<>'',IF{SQL{SELECT 1 FROM RMPROP WHERE RMPROPID = '[PROPERTYID]'}=1,0,-1},-1},You must provide a valid PropertyID.}

   Note

   The **VALIDATE{}** expression has a condition and an error message that displays when the condition is true. The condition is true when the result is -1. For more information on the VALIDATE{}, IF{}, and SQL{} expressions, refer to the *WebDesign User Guide*.

8. Repeat Steps 5 through 7 for the **BUILDINGID** input parameter, with the following expression:

   VALIDATE{IF{[BUILDINGID]=[*] AND '[BUILDINGID]'<>'',IF{SQL{SELECT 1 FROM RMBLDG WHERE RMBLDGID = '[BUILDINGID]'}=1,0,-1},-1},You must provide a valid BuildingID.}

9. Repeat Steps 5 through 7 for the **UNITID** input parameter, with the following expression:

   VALIDATE{IF{[UNITID]=[*] AND '[UNITID]'<>'',IF{SQL{SELECT 1 FROM UNIT WHERE UNITID = '[UNITID]'}=1,0,-1},0},You must provide a valid UnitID.}

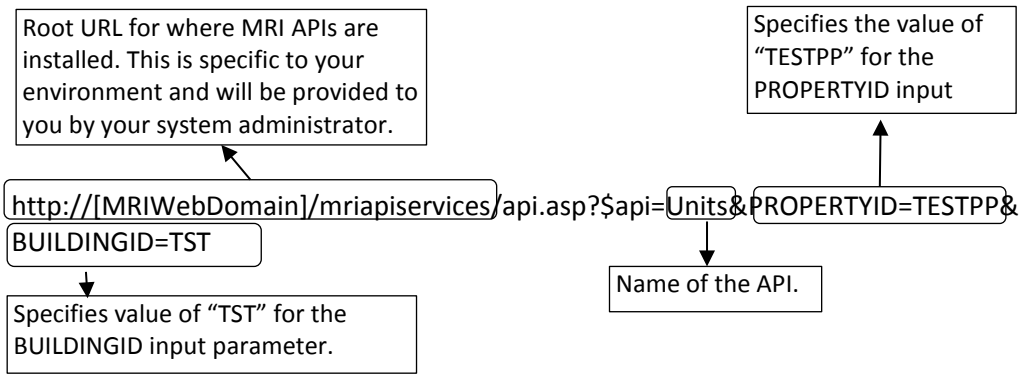
   Note

   Because **UNITID** is an optional parameter, if the parameter is not passed in, 0 is returned to the VALIDATE{} expression and the error message is not displayed.

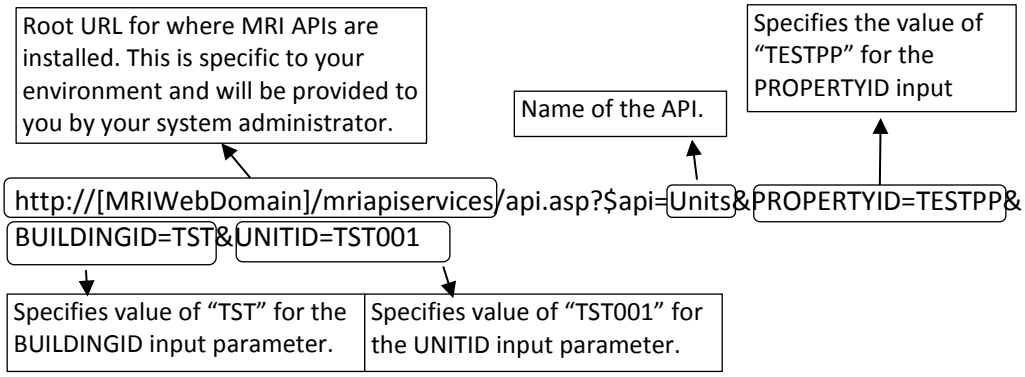
Testing the API

To test the Units API in any browser, follow these steps:

1. Give your Web Services role access to your API, as described in the *System Administration Guide*. For this scenario, in the **Access** column, select the check box next to the Units API.
2. To test leaving the optional filter blank, open any browser and go to the following URL for the Units API:

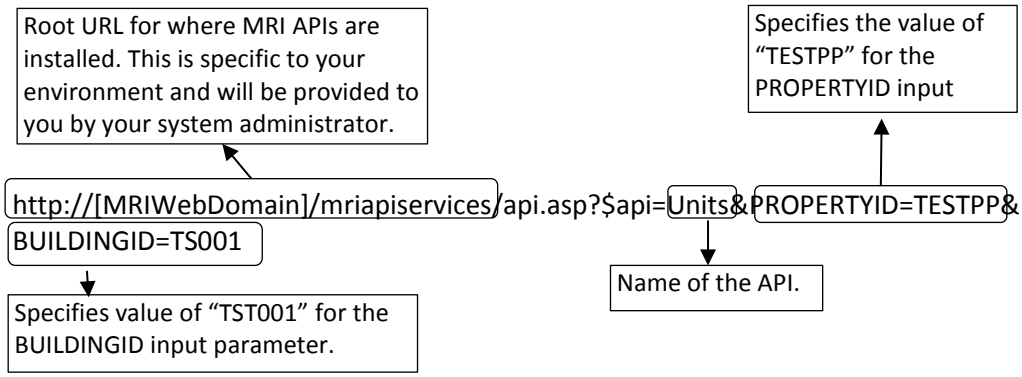


3. Enter your credentials, as described in "Testing Read-Only APIs" on page 11.
4. Review the results. You should see all units for the property specified in the query string and the building specified in the query.
5. To test functionality of the optional filter, open any browser and go to the following URL for the Units API:



6. Enter your credentials, as described in "Testing Read-Only APIs" on page 11.
7. Review the results. You should only see units from the property and building specified in the query.
8. To test the validation of the required fields when left blank, open any browser and go to the following URL for the Units API:

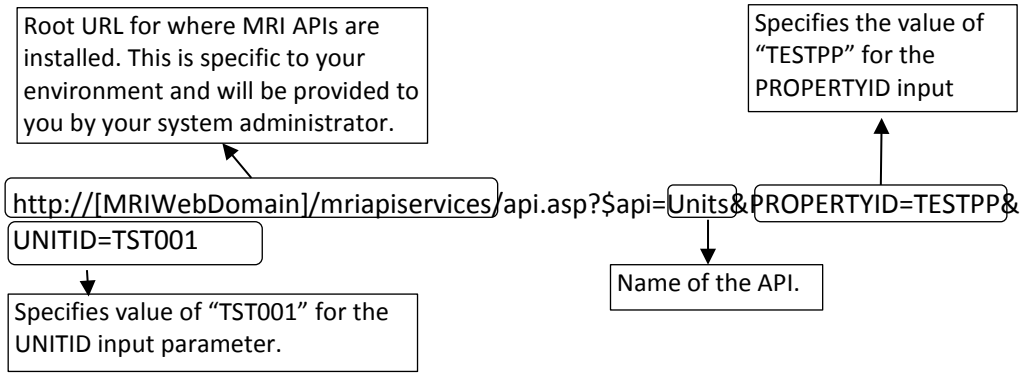
9.



10. Enter your credentials, as described in "Testing Read-Only APIs" on page 11.

11. Review the results. You should receive the message that states, "You must provide a valid BuildingID," and no results should be returned.

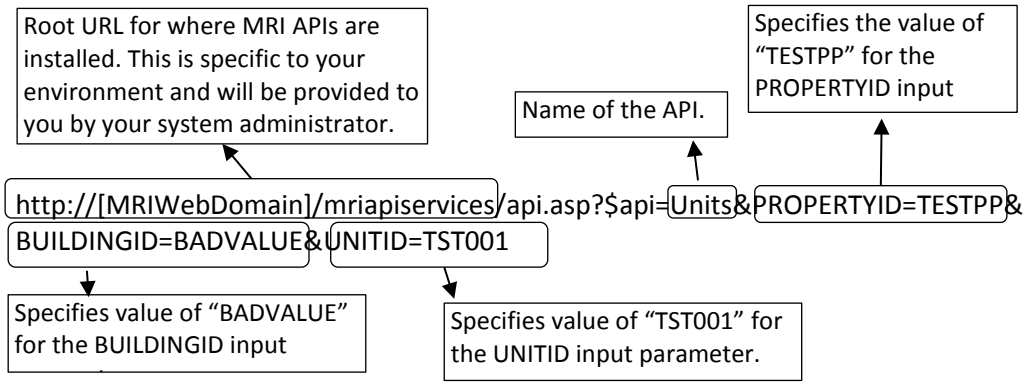
12. To test the validation of the required fields when left blank, open any browser and go to the following URL for the Units API:



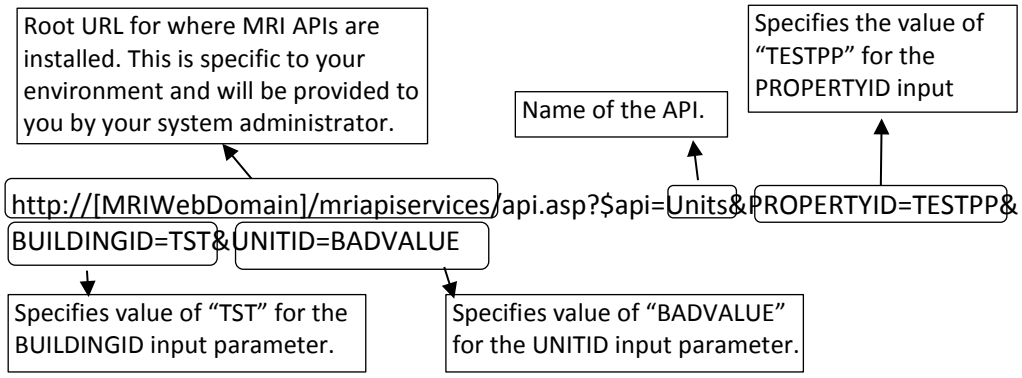
13. Enter your credentials, as described in "Testing Read-Only APIs" on page 11.

14. Review the results. You should receive the message that states, "You must provide a valid PropertyID," and no results should be returned.

15. To test the validation of the required fields when an incorrect value is passed in, open any browser and go to the following URL for the Units API:



16. Enter your credentials, as described in "Testing Read-Only APIs" on page 11.
17. Review the results. You should receive the message that states, "You must provide a valid BuildingID," and no results should be returned.
18. To test the validation of the optional field when an incorrect value is passed in, open any browser and go to the following URL for the Units API:



19. Enter your credentials, as described in "Testing Read-Only APIs" on page 11.
20. Review the results. You should receive the message that states, "You must provide a valid UnitID," and no results should be returned.

Scenario 9: Improving Performance of an API thatChanges Data in a Table.

In this scenario, you will modify the Vendors API from Scenarios 1 and 3. Before you begin, make sure you have completed these scenarios, as described in "Scenario 1: A Filterable API that Reads Data from a Single Table" on page 13 and "Scenario 3: An API that Changes Data in a Table" on page 16. This scenario will improve performance by

limiting the records that need to be loaded from the database to just the records that the caller intends to update.

Building the API

To add to the Vendors API from Scenario 3, follow these steps:

1. Open the Vendors API.
2. Add a Meta field, as described in "Understanding Meta Fields" on page 6.
3. In the **VariableName** property, enter **META.INCOMING.[GridID].[PrimaryKey]**. For this scenario, enter **META.INCOMING.VEND.VENDID**.
4. Right-click the grid, click **Display Properties**, and then click the **Properties** tab.
5. Double-click the **Query** property, and then enter the following expression:

   VENDID IN ([META.INCOMING.VEND.VENDID])

Testing the API

This API cannot be tested in a browser. MRI Software recommends using an HTTP debugging proxy server software. If your company does not allow you to install proxy software, you can search for “HTTP” in your browser’s online store to find a browser application or extension to help you issue HTTP requests to your API. To test this API, follow these steps:

1. Set up two new vendors in MRI, and set the vendor IDs to **VEND1** and **VEND2**. For instructions on setting up a vendor type, refer to the “Managing Vendor Types” topic in the web Help (**Accounts Payable > Setup & Maintenance**)
2. Update the following request with your information, and then use your HTTP tool to issue the request:

   Note

   For more information on how to invoke APIs that change data, refer to the *API Integrations Guide.*

```
PUT http://[MRIWebDomain]/mriapiservices/api.asp?$api=[API
    name] HTTP/1.1
Authorization: Basic [AuthorizationString]
Content-Type: application/xml
Accept: application/xml
Content-Length: [depends on size below]

<?xml version="1.0" encoding="utf-8"?>
<Vendors>
  <entry>
    <VendorID>VEND1</VendorID>
    <VendorName>Updated Vendor Name 1</VendorName>
    <City>Cleveland</City>
    <State>Ohio</State>
```

```
    <Zip>44444</Zip>
    <Status>A</Status>
    <Contact>Sample Name 1</Contact>
  </entry>
  <entry>
    <VendorID>VEND2</VendorID>
    <VendorName>Updated Vendor Name 2</VendorName>
    <City>Cleveland</City>
    <State>Ohio</State>
    <Zip>44444</Zip>
    <Status>A</Status>
    <Contact>Sample Name 2</Contact>
  </entry>
</Vendors>
```

3. Verify that you received a response with HTTP status code of 200. You should see the updated vendor information in MRI. For this scenario, you should see Updated Vendor Name 1 and Updated Vendor Name 2. For more information on creating or modifying data with APIs, refer to the *API Integrations Guide.*

Scenario 10: An API That Uses Query String Values While Changing the Data

In this scenario, you will modify the Vendors API from Scenarios 1 and 3. Before you begin, make sure you have completed these scenarios, as described in "Scenario 1: A Filterable API that Reads Data from a Single Table" on page 13 and "Scenario 3: An API that Changes Data in a Table" on page 16. This scenario will enhance the API to allow the caller to specify a query string parameter to set a single Vendor type for all records that are being changed.

Building the API

To add to the Vendors API from Scenario 3, follow these steps:

1. Open the Vendors API.
2. Add input parameters, as described in "Setting Up Input Parameters" on page 11. For this scenario, add one input parameter and set the **VariableName** property to **VENDORTYPE**.
3. Add a column to the grid, as described in "Adding Columns to Your Grid" on page 9. For this scenario, drag and drop the VENDTYPE table field into the grid on the page.
4. Review the **ColumnName** properties for each column and revise as necessary, as described in "Revising the ColumnName Property" on page 10.
5. To set the vendor type, add a pre-insert trigger to the grid by following these steps:

a. Right-click the grid, and then click **Display Properties** to open the **Grid Properties** dialog box.

b. On the **Properties** tab, double-click the **PreInsertTrigger** property.

c. In the **Expressions** dialog box, enter the following expression:

   SETFLD{VEND.VENDTYPE, [VENDORTYPE]}

d. Click **Ok**.

Testing the API

This API cannot be tested in a browser. MRI Software recommends using an HTTP debugging proxy server software. If your company does not allow you to install proxy software, you can search for "HTTP" in your browser's online store to find a browser application or extension to help you issue HTTP requests to your API. To test this API, follow these steps:

1. Set up a vendor type called **VTYPE1** in MRI. For instructions on setting up a vendor type, refer to the "Managing Vendor Types" topic in the web Help (**Accounts Payable > Setup & Maintenance**)
2. Update the following request with your information, and then use your HTTP tool to issue the request:

Note

For more information on how to invoke APIs that create data, refer to the *API Integrations Guide.*

```
POST http://[MRIWebDomain]/mriapiservices/api.asp?$api=[API
       name]&VENDORTYPE=VTYPE1 HTTP/1.1
Authorization: Basic [AuthorizationString]
Content-Type: application/xml
Accept: application/xml
Content-Length: [depends on size below]

<?xml version="1.0" encoding="utf-8"?>
<Vendors>
  <entry>
    <VendorID>VEND1</VendorID>
    <VendorName>Updated Vendor Name 1</VendorName>
    <City>Cleveland</City>
    <State>Ohio</State>
    <Zip>44444</Zip>
    <Status>A</Status>
    <Contact>Sample Name 1</Contact>
  </entry>
  <entry>
    <VendorID>VEND2</VendorID>
    <VendorName>Updated Vendor Name 2</VendorName>
    <City>Cleveland</City>
    <State>Ohio</State>
    <Zip>44444</Zip>
    <Status>A</Status>
```

```
    <Contact>Sample Name 2</Contact>
  </entry>
</Vendors>
```

3. Verify that you received a response with HTTP status code of 200. You should see the updated vendor information with the new vendor type. For more information on creating or modifying data with APIs, refer to the *API Integrations Guide.*

Product Codes

Product Name	Code
Accounts Payable	AP
Budgeting and Forecasting	BF
Commercial Management	CM
Corporate Accounts Receivable	CAR
Corporate Real Estate	CRE
General Ledger	GL
JobCost	JC
Lease Flow	LF
Residential Management	RM
Viewpoint	AM

Shaping Responses

When sending data into the API, use the Content-Type header to indicate the shape of the data that you are sending, and use the Accept header to indicate the shape of the data that you want returned. For information on how to learn about the structure of an API without invoking it, refer to the API Integrations Guide.

You can specify that you want your response formatted as AtomPub, JSON, Plain XML, or CSV. If transformation is not specified, OData AtomPub feed is the default response.

AtomPub

You can request AtomPub in the following ways:

- Send an Accept: application/atom+xml header.
- Append the $format=atom query string parameter.

JSON

You can request JSON in the following ways:

- Send an Accept: application/json header.
- Append the $format=JSON query string parameter.

Plain XML

You can request plain XML in the following ways:

- Send an Accept: application/xml header.
- Append the $format=xml query string parameter.

CSV

You can request CSV in the following ways:

- Send an **Accept: text/csv** header.

  Note

  If you want to omit CSV headers, send an **Accept: text/csv;header=absent** header.

- Append the **$format=csv** query string parameter.