

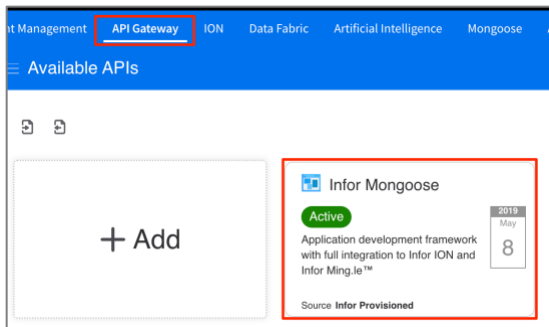


# Infor OS App Designer Guide to Mongoose Data Services

This guide will walk you through the process of creating a data service in Infor OS App Designer that connects to the Mongoose APIs.

## Prerequisites

- Verify that the Infor Mongoose suite has been provisioned in the tenant. Under API Gateway, verify that the Infor Mongoose suite is active:



- A strongly typed API has already been generated and is available to access within the API Gateway. For instructions on how to use the REST API Wizard within Mongoose, reference: [KB-2294576](#)
- Verify that the following two endpoints exists:

Available APIs / Infor Mongoose

Application development framework with full integration to Infor ION and Infor Ming.le™

Endpoints

	Endpoint	Description	Indexing Status	Documentation
<input type="checkbox"/>	MONGOOSE/IDORestService/MGRestService.svc	Mongoose REST Service		
<input type="checkbox"/>	MONGOOSE/IDORestService/ido	Mongoose REST Service V2		
<input type="checkbox"/>	MONGOOSE/IDORestService/ido/dynamic	Mongoose REST V2 Swagger...		

Page 1 of 1

- The name of the Mongoose configuration you want to access.
- Ensure that the user has the following security roles added in OS Portal before proceeding:
  - IAD-DataMaintainer
  - IAD-Designer
  - IAD-Publisher

# Step 1: Identify and Test the API in the Infor API Gateway

Testing the data in the API Gateway is an optional step, however it is good to perform this step just to verify that the API is working as expected before creating a data service.

- From the OS Portal landing page, select **API Gateway**.
- Select the **Infor Mongoose** suite.
- Generally speaking, there are two types of APIs that can be used with OAD:
  - **Strongly Typed API (recommended)**: This is an API endpoint that has been dynamically created using the REST API Wizard within Mongoose. This API can have the collection, properties, and parameters all predefined eliminating the need for additional configuration when creating a data service in OAD.
  - **Standard Load Collection API**: This is a standard API that can be used to call a specific collection, properties, and filtering parameters. Using this method would require a more in-depth knowledge of the data itself and requires more configuration when configuring a data service in OAD.

## Testing a Strongly Typed API

- Click the documentation icon for the **MONGOOSE/IDORRequestService/ido/dynamic** endpoint.
- Click the documentation icon for the strongly typed API that you created using the REST API Wizard as stated in the prerequisites. In this example, we are going to reference the METCustomers business class that was dynamically created in my environment.
- Navigate to the **GET /api-docs-collection/METCustomers/GetCustomerList** API. In the X-Infor-MongooseConfig parameter, specify the configuration name. Leave the default values for ido and properties.

The screenshot shows the 'Available APIs' section in the Infor API Gateway. It displays a table with the following parameters:

Name	Description
X-Infor-MongooseConfig (required)	Mongoose configuration to log into; required when using the service through KIN API, not needed otherwise. <b>STARGATE_MONGOOSE_DEM_DEFAULT</b>
ido (required)	IDO name METCustomers
properties (optional)	A comma-delimited property list. Provide * to include all, except subcollection properties. *

At the bottom, there is an 'Add Item' button and a 'Filter' section with 'SQL filter string'.

- ```
Code    Details

200      Response body

{
  "Items": [
    {
      "Address": "310 Toban Court",
      "ARCredit": "25000.00",
      "City": "Waco",
      "Contact": "Andrew Wells",
      "ContactEmail": "awells@aura.com",
      "Country": "United States",
      "CreateDate": "20221027 15:07:25.927",
      "CreatedBy": "caitlin.porter@infor.com",
      "CreditLimit": null,
      "InWorkFlow": "0",
      "Logo":
        "iVBORw0KGgoAAAANSUHEUGAAAFQAAH0CAYAAADLIt+KAAAAAXNSRQ
        MEIwdQVoahUgRvncmTEk5Y3prTzIkZ8-Cjx4Dnhtcg1GdESRG6
        Zho0ARlyE93abCudeMab3JnrvnLEZ50TRwMDlWMjIcmRlXNSBFR
        LZ8j12V5ZWllBrnRzlZeUMS8nC1ApeGi1sbMrvcneG1np3JsddRW8t5
        TESMDMSZyxNDY0NDQzcP9yZGV6BgkCiAgIDYvcnRmOmlNLrvcNtk4
        94ybAGQj1YXRvc1RvbWZ+CiaRL3kJzjPNZnjcmldwGlvbj4KPcy9Z
        U2l2Pvc0SE3Ec+S5kSXZlSVbezaCS7BaMDzBgLewdBkQX88qP1RQ
        KaphocvrnP81Mu0exZ95z79hjTHHNJF/Q19AJ0CPQI9AJ0CNz4CL
        bkEEQdgfgPYDFGHFrvcnoEegRGBHoEegA3a/BnoEGRGBHoEegr00Q
        iUEFFoCGPDRGFUF5-PCBFHfZMLfSYvGbf5-PCGFHfZMLfSYvGbf5-
```

## Testing an Advanced Load Collection Request

- From the OS Portal landing page, select **API Gateway**.
- Select the **Infor Mongoose** suite.
- Click the documentation icon for the **/STARGATE\_DEM/MONGOOSE/IDORequestService/MGRestService.svc** endpoint.
- Navigate to the **GET /json/{ido}/{props}/adv** API. This API is an advanced load collection request, that will allow the user to add in many different input parameters. Select the API, and then click the **Try it out** button.
- Specify the Mongoose configuration, the IDO name, the properties you want to retrieve, and optionally a filter value and/or orderby value:

GET /json/{ido}/{props}/adv LoadCollectionAdvanced

A Load Collection with all potential parameters available

Parameters

| Name                                                     | Description                                                                                                                                    |
|----------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Authorization<br><small>string (header)</small>          | Token obtained through a call to SecurityToken, or if using the service through ION API, a valid Authorization - Token obtained through a call |
| X-Infor-MongooseConfig<br><small>string (header)</small> | Mongoose configuration to log into; required when using the service through ION API, not needed otherwise                                      |
| ido * required<br><small>string (path)</small>           | IDO name                                                                                                                                       |
| props * required<br><small>array[string] (path)</small>  | Comma-separated property names                                                                                                                 |
| filter<br><small>string (query)</small>                  | SQL filter string                                                                                                                              |
| orderby<br><small>string (query)</small>                 | SQL ORDER BY value                                                                                                                             |
| rowcap                                                   | Sets row cap: -1 = default in Mongoose; 0 = unlimited.                                                                                         |

Values entered (highlighted with red boxes):

- X-Infor-MongooseConfig: STARGATE\_MONGOOSE\_DEM\_DEFAULT
- ido: METCustomers
- props: CustNum, Name
- orderby: CustNum

- Click the blue **Execute** button. View the response body below. A code of 200 means that it was able to successfully retrieve data:

200

Response body

```
{
  "Bookmark": "<B><P><p>CustNum</p></P></D></f></D>",
  "Items": [
    [
      {
        "Name": "CustNum",
        "Value": "1865"
      },
      {
        "Name": "Name",
        "Value": "Aura Cafe & Restaurant"
      },
      {
        "Name": "_ItemId",
        "Value": "PBT=[METCustomers] CUST_DT=[2022-12-07 10:00:00]"
      }
    ],
    [
      {
        "Name": "CustNum",
        "Value": "1870"
      },
      {
        "Name": "Name",
        "Value": "Aura Cafe & Restaurant"
      },
      {
        "Name": "_ItemId",
        "Value": "PBT=[METCustomers] CUST_DT=[2022-12-07 10:00:00]"
      }
    ]
  ]
}
```

## Step 2: Create the Data Service in Infor OS App Designer

Now that we have successfully retrieved the data we need in the API Gateway, we can now create a data service that will call this same API.

### Creating a Strongly Typed Data Service

- Click on App Designer from the application ribbon within OS Portal.
- Click Data Services from the left pane and create a new data service.
- Select **ION API** for the type.
- Search for **GetCustomerList** and select **Get** for the method type. Select the API that we tested earlier:

Create Data Service - Find API

Search:  Suite:  Methods:

| Name            | Rest Path                    | Proxy Path             | Details                 | Select                 |
|-----------------|------------------------------|------------------------|-------------------------|------------------------|
| GetCustomerList | /api-docs-collection/METC... | STARGATE_DEM/MONGOO... | <a href="#">Details</a> | <a href="#">Select</a> |

- Give the data service a name and click **Next**.
- Next to the **Request Header** field, click the pencil icon to add in a request header parameter. Add in the **Key** of **X-Infor-MongooseConfig** and the **Value** should be the Mongoose configuration name you want to access:

Request Header Editor

+

| Key                    | Value                         | Actions                               |
|------------------------|-------------------------------|---------------------------------------|
| accept                 | application/json              | <input type="button" value="Delete"/> |
| X-Infor-MongooseConfig | STARGATE_MONGOOSE_DEM_DEFAULT | <input type="button" value="Delete"/> |

- Because the IDO name and properties is already hardcoded into this strongly typed API, there is no need to specify those default values. Under the **Input Parameters**, click the trash can icon on the right to delete the **ido** and **properties** parameters.
- Optionally, you can specify a filter, orderby, recordcap, etc.:

- Click the **Test Data Service** button, and then click the **Run Data Service** button at the bottom to verify a successful connection:

- Click **Next**. Change the root element `$.Items[*]` and add in the two output parameters as seen below. *It is important to note that this root element might change depending on the type of API you selected.*

Create Data Service

Progress: Name & Description → Input → **Output** → Summary

Basic Request Information

Format: **JSON** [VIEW JSON](#)

Root Element: **\$.items[\*]** Object Id:

Output Parameters

[Generate parameters](#) [JSON Help](#) [+ Add Parameter](#)

| Name    | Alias   |                          |
|---------|---------|--------------------------|
| CustNum | CustNum | <input type="checkbox"/> |
| Name    | Name    | <input type="checkbox"/> |

[Test Data Service](#)

Previous Next Finish

- Click the **Test Data Service** button, and then click the **Run Data Service** button at the bottom to verify the output parameters are correct:

Test Data Service

loadType:

bookmark:

pgc:

readOnly:

[Run Data Service](#)

Response Body

| Name    | Description            |
|---------|------------------------|
| CustNum | 1065                   |
| Name    | Aura Cafe & Restaurant |

[OK](#)

- Click **Ok** then click **Next** and **Finish**. Your data service is now ready to be used when building widgets within Infor OS App Designer.

## Creating a Standard Load Collection Data Service

- Click on **App Designer** from the application ribbon within OS Portal.
- Click on **Data Services** from the left pane and then click the Create a new data service button.
- Select **ION API** for the type.
- Search for **LoadCollectionAdvanced**, select **Infor Mongoose** as the suite, and **Get** for the method. Select the following API:

Create Data Service - Find API

Search:  Suite:  Methods:

| Name                   | Rest Path               | Proxy Path             |                         |                        |
|------------------------|-------------------------|------------------------|-------------------------|------------------------|
| LoadCollectionAdvanced | /xml/{ido}/{props}/adv  | STARGATE_DEM/MONGOO... | <a href="#">Details</a> | <a href="#">Select</a> |
| LoadCollectionAdvanced | /json/{ido}/{props}/adv | STARGATE_DEM/MONGOO... | <a href="#">Details</a> | <a href="#">Select</a> |
| LoadCollectionAdvanced | /xml/{ido}/adv          | STARGATE_DEM/MONGOO... | <a href="#">Details</a> | <a href="#">Select</a> |
| LoadCollectionAdvanced | /json/{ido}/adv         | STARGATE_DEM/MONGOO... | <a href="#">Details</a> | <a href="#">Select</a> |

- Give the data service a name and click **Next**.
- Next to the **Request Header** field, click the pencil icon to add in a request header parameter. Add in the **Key** of **X-Infor-MongooseConfig** and the **Value** should be the Mongoose configuration name you want to access.

Request Header Editor

+

| Key                    | Value                         | Actions                               |
|------------------------|-------------------------------|---------------------------------------|
| accept                 | application/json              | <input type="button" value="Delete"/> |
| X-Infor-MongooseConfig | STARGATE_MONGOOSE_DEM_DEFAULT | <input type="button" value="Delete"/> |

- In the **Input Parameters** section, add in the default IDO collection name, as well as any additional parameters you want to include such as a filter, rowcap, orderby, etc.

Create Data Service

GET `json/{ido}/{props}/adv?filter={filter}&orderby={orderby}&rowcap={rowcap}`

Request Header

accept:application/json  
X-Infor-MongooseConfig:ADCQA\_Mongoose\_TST\_DEFAULT

VIEW JSON

Input Parameters

JSON Help + Add Parameter

| Required                            | Name    | Type  | Alias (Optional) | Default (Optional) |   |
|-------------------------------------|---------|-------|------------------|--------------------|---|
| <input checked="" type="checkbox"/> | ido     | Path  |                  | METCustomers       | 🗑 |
| <input checked="" type="checkbox"/> | props   | Path  |                  | CustNum_Name       | 🗑 |
| <input type="checkbox"/>            | filter  | Query |                  | CustNum=1065       | 🗑 |
| <input type="checkbox"/>            | orderby | Query |                  | CustNum desc       | 🗑 |
| <input type="checkbox"/>            | rowcap  | Query |                  | 10                 | 🗑 |

Previous Next Finish

Make sure you remove any input parameters that you are not using by clicking the trash can icon on the right.

- Click the **Test Data Service** button, and then click the **Run Data Service** button at the bottom to verify a successful connection:

Test Data Service

CustNum desc

rowcap

10

Run Data Service

Response Body

```
{
  "Bookmark": "<B><P><P><CustNum</p></p><D><B>true</B></D><F><V>1065</V></F><L><V>1065</V></L></B>",
  "Items": [
    {
      {
        "Name": "CustNum",
        "Value": "1065"
      },
      {
        "Name": "Name",
        "Value": "Aura Cafe & Restaurant"
      },
      {
        "Name": "_ItemId",
        "Value": "PBT[METCustomers] CUST.DT=[2023-03-27 19:11:25.793] CUST.ID=..."
      }
    }
  ]
}
```

OK

- Click **Next**.
- In this case, the **Root Element** will need to be changed to **\$.Items[\*]** because of this particular APIs JSON format. *It is important to note that this root element may change if you choose to use a different API.*
- Using the **Add Parameter** button, add in each output parameter you want in the format demonstrated below:

Create Data Service

Progress: Name & Description (selected), Input, Output, Summary

⚠ Limited Documentation: Type is missing in schema, review your swagger file. X

Basic Response Information

JSON VIEW JSON

Root Element: \$.items[\*] Object Id:

Output Parameters

Generate parameters JSON Help + Add Parameter

| Name            | Alias   |                      |
|-----------------|---------|----------------------|
| <u>.0.Value</u> | CustNum | <input type="text"/> |
| <u>.1.Value</u> | Name    | <input type="text"/> |

Test Data Service

Previous Next Finish

- Use the **Test Data Service** button at the bottom to ensure the output parameters piece is working as expected.

Test Data Service

props: CustNum, Name Required

filter: CustNum=1065

orderby: CustNum desc

rowcap: 10

Run Data Service

Response Body

| Name    | Description            |
|---------|------------------------|
| CustNum | 1065                   |
| Name    | Aura Cafe & Restaurant |

OK

- Click **Next** and then click **Finish**. Your data service is now ready to be used when building widgets within Infor OS App Designer.