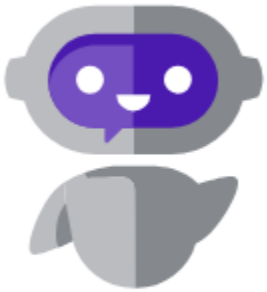


TEAM CENTRAL USER GUIDE

Legal Notices

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted to parties other than the customer and the customer's employees in any form or by any means, electronic or mechanical, for any purpose, without the express written consent of TeamCentral.

This document contains confidential proprietary information and trade secrets of TeamCentral. This document is distributed with the understanding that it will not be disclosed to any third party, in whole or part, without the prior written consent of TeamCentral.

Contact Information

Address: TeamCentral, Inc
1215 Lyons Rd Bldg F
Dayton, OH 45458-1828

Email: support@teamcentral.ai

User Guide Change Log

Version	Date	Description of Changes
v1.1	10/25/2024	Updates to Monitor (Error Logs, Record History, Identifiers); On-Demand Publisher & Scheduler icon; Version control added.

Table of Contents

Legal Notices 2
Contact Information 2
User Guide Change Log 3
Introduction to this Guide 6
About Central 6
Purpose of this Guide 6
Getting Started Checklist 6
Good to Know Definitions 7
User Guide Conventions 8
Introduction 9
The Process 9
Organizing Your Data 9
Define Schemas 10
Configure Connectors 10
Build Data Hubs 11
Deploy and Monitor 12
Build and Manage Schema Definitions 13
Schema Types 13
Simple Schema Definition 13
Foreign Key Schema Property 13
List Schema Property 13
Transaction Types 13
Add a Schema Definition 14
Edit a Schema Definition 18
Delete a Schema Definition 20
Host Type Extension Schema Definition 21
Add a Host Type Extension Schema Definition 21
Edit or Delete a Host Type Extension Schema Definition 23
Build and Manage Connectors 24
Add a New Connector 24
Edit a Connector 25
Delete a Connector 26
Connect a Connector 27
Test a Connector 28
Build and Manage Data Hubs 29
Build a Data Hub 29

Add Endpoints to a Data Hub 30
Configure Endpoints 33
Endpoint Action Inputs 35
Parameterizing 37
Key Scraping Action 38
Endpoint Relationships 39
Add a Child Endpoint 40
Add a Dependent Publisher 41
Map Endpoint Actions 43
Mapping Properties of Endpoint Actions per Protocol 47
Adapting Data when Mapping 50
Advanced Tab 50
Transforms Tab 51
Behaviors Tab 53
Additional Data Hub Adaptation Methods 54
Publish/Subscribe Filters 54
Scheduler 56
Variables 59
Drop-Down Lookups 60
View Code 62
Versioning 63
Deployment 64
Deploy a Data Hub 64
Monitor a Data Hub 66
View the Dashboard 66
View Error Logs 69
View Message Search 71
Message Search Reference Table 73
Central Common Model Definitions 74
Appendix A: Connectors Catalog 76

Introduction to this Guide

About Central

Central is quick to implement, easy to scale, highly performant, and no-code to maintain! It is the next generation in automation technology, pairing a low/no-code integration platform with a data quality management solution to help data synchronize across multiple systems.

To fully leverage data as an asset that powers efficiency gains, competitive advantage, business growth, and profitability, companies need to streamline the way data is managed across end-to-end processes. Every one of your business' teams relies on each other to make the company successful by consuming and sharing data; but in many cases, that data is not easy to access or manage across departments utilizing multiple specialized systems.

Other no-code options were not solving the entire process that a software developer goes through to build a solution – design, build, test, version control, deployment, monitoring, maintenance. Likewise, and more importantly, the other options imparted no emphasis on the most important asset to the company – the actual data – including the quality of the data, managing metadata and the relationships within your data, and governing your data architecture, to name a few. We believe Central provides this solution!

Central addresses your most complex data synchronization and workflow needs across eCommerce, ERP, CRM, HRIS, Purchasing, and Supply Chain systems — with minimal coding required. Central's automation empowers businesses to continue scaling while keeping personnel costs in check. It streamlines workflows, increases productivity, and boosts the bottom line by connecting systems and automating dataflow.

Purpose of this Guide

This guide is intended to assist users and developers in building and customizing your company's version of Central software. It is assumed the person working to create your version of Central has a technical understanding of database structure, data terminology, your business solutions, as well as data inputs and downstream consequences. Of course, we are always here to help, so feel free to reach out if you have any questions before you get started (support@teamcentral.ai).

Getting Started Checklist

Ensure you have the following before beginning to build your model.

- ☐ Central software URL
- ☐ Central Admin access
- ☐ Working knowledge of the software systems you plan to integrate
- ☐ Technical documentation for each connecting software system, including the selector authentication information

Good to Know Definitions

Terminology	Definition
Common Model	This is the starting point or guide for defining common entities within your business, e.g. Customer, Vendor, Employee, Item, Sales Order, etc. You will map data from your source systems to the Common Model. All publishing and subscribing will flow through the Common Model, allowing for complex data interactions within a more simplistic integration product.
Schema Definition	This is your customized copy of the Common Model – defining and detailing the data you need to flow between systems (e.g. Customer).
Schema Property	Schema Properties are components that make up the Schema Definition (e.g. a Schema Definition of Customer may have Schema properties that include name, address, contact, etc.)
Connector	Connectivity to any data source (e.g. SaaS, API, Internal Customer API, Database, File Store, etc.)
Endpoint	Specific instance of a Connector for the type of data you want to connect to (e.g. Salesforce Account Data)
Endpoint Action	A specific set of instructions on how the Endpoint will perform tasks such as reading, writing, and deleting data on the Endpoint.
Publisher	An Endpoint that triggers data to be integrated.
Subscriber	An Endpoint that receives data to be integrated.
Data Hubs	A logical grouping of Endpoints that make up a single data synchronization solution.
Agent	At the center of all transactions, an Agent is constantly pushing data out and receiving data in.

User Guide Conventions

Throughout this user guide, you will encounter the following note boxes and callouts, documenting additional information, helpful tips, examples, and best practices.

NOTE:	This is useful information or a helpful tip regarding the current topic.

NOTE:	This is **critical** information regarding the current topic.

Advanced Tool:	Descriptions of advanced features within the software, intended for those with developer knowledge and skills.

Advanced Configuration:	Specific implementation examples of advanced features within the software, intended to give developers a broader view of atypical applications.

Example:	This callout will reference analogies and/or examples for added illustration and understanding.

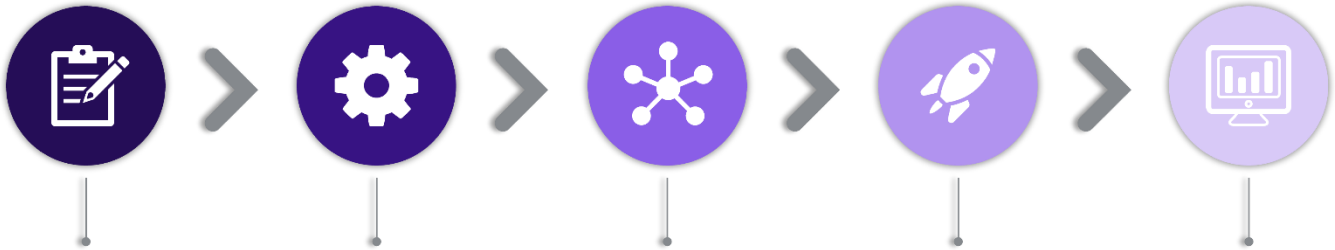
Best Practice:	This callout will recommend best practices on the software set-up and maintenance.

Introduction

Central starts with a Common Model, which serves as a platform for you to build customized, secure, and reliable connections associating data from one system to another, and what data is connected is determined specifically by your business needs. It is important to have a strategy to ensure implementation is quick, efficient, and successful. You will need to assess your business systems, align on your specific business needs, and strategically map out your version of Central.

The Process

Upon logging into Central for the first time, you'll need to correlate your data to the Common Model, enabling it to interact with each connected system. The process starts by defining Schemas (data values) and identifying Connectors (systems) that store those Schemas. Then, using your defined business strategy and knowledge about how the data should flow, you'll build Data Hubs, organized around logical groupings (e.g. Customer Billing Data, Employee Personal Data, Sales Leads, etc.). Finally, you can test and deploy each Data Hub to start the transfer of data between systems. Once you've successfully deployed your Data Hub(s), you can monitor the status of each to better understand how often the data transfers take place, if there were any errors in data transfer, and the overall success of the integration.



Define Schemas
Define your data by setting up a Schema Definition for each data point you'd like to connect.

Configure Connectors
Select and configure the systems that will subscribe and/or publish data to the Common Model.

Build Data Hubs
Design the flow of data between the Common Model and Connectors by grouping logical and precise business events within a single hub.

Deploy
Deploy your Data Hub(s), enabling data to be read and written across systems.

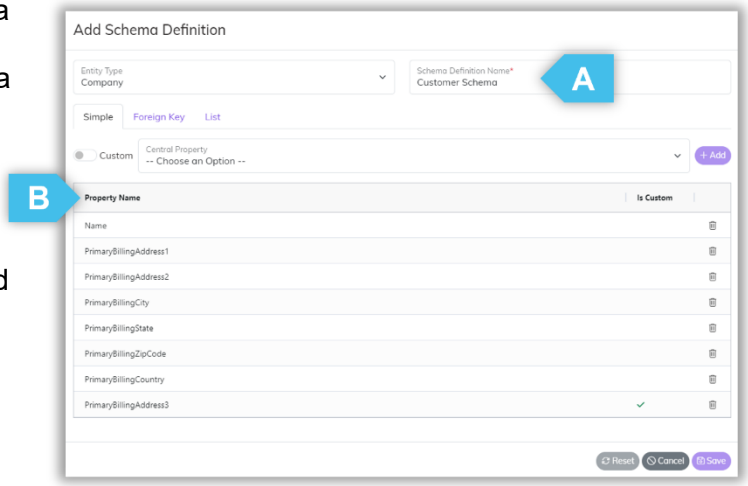
Monitor
Monitor the status of your Data Hubs and the health of the integration.

Organizing Your Data

To start, it is advised to organize the data you'd like to connect between systems. To do this, you'll need to understand and document what data is captured by each system and what data you'd like to flow between those systems. A good way to start is to document the data within each system that you want to connect. For example, if your source of truth for Customer data is Salesforce, document all of the Customer fields that you want to include, such as *Name*, *Address*, *City*, *State*, *Zip Code*, *Phone Number*, etc. Then, go to each system that you plan to connect to Salesforce and document the data fields corresponding to those same Customer fields. You will end up with a cross-reference of all Customer data from each of the systems that you're planning to connect. Completing this step will help you more easily and quickly define Schemas, build Connectors, and map Endpoints. So, any field that you plan to read or write data to or from needs to be documented, as it will become a Schema Property within your Schema Definitions.

SCHEMA DEFINITION: **CUSTOMER**		
System 1	**System 2**	**System 3**
FirstName	FirstName	Name_First
LastName	LastName	Name_Last
NickName		Name_Nick
CustomerID		ID_Customer
Address1	Address1	Address_1
Address2	Address2	Address_2
City	City	City
State	State	State
ZipCode	Zip	Code_Zip
Phone1	Phone	Phone_Home
Phone2		Phone_Cell
Email	Email1	Email
	Email2	

Define Schemas



Schemas are your customized definitions for the data you are trying to replicate between systems. You'll start by selecting an Entity Type (and as applicable, a Sub-Type) to identify the category of data; and then you'll name (A) the data you'd like to capture, defining it as a Schema. A Schema Definition is simply an identifier that describes your data, such as contact person. When you start Mapping data between Endpoints (or systems), it will all be mapped around your Schema Definitions.

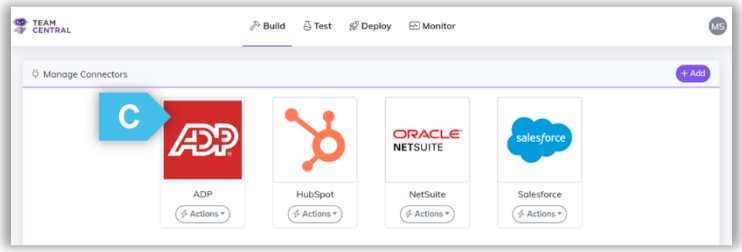
Within a Schema Definition, you'll assign Schema Properties (B) that detail the data for that Schema Definition. For example, a Schema Definition of Contact Person may have Schema Properties that include name, email, phone number, customer address, etc. There are three (3) types of Schema Properties you can add to your Schema Definition, each with their own defined purpose.

A Simple Property is a single value to single value property. It represents data that is both common and static, meaning it applies to all (or most) of your data systems and it isn't likely to change within the systems you're synchronizing, such as a customer name.

Foreign Key Properties are references to a data (or value) that may be different from one system to the next. They are cross-referenced values that need to be converted to the expected value for each subscribing system, such as a customer record number in your sales system vs your billing system. These definitions are customer-built and customized to your specific needs.

List Properties are used for hierarchy (parent/child) customizations, such as a record that contains multiple components. For example, a sales order has a header with customer data, as well as line items for purchased components – all data you may want to capture within a List Schema.

Configure Connectors



Connectors are how data gets transferred from one system to the next. They provide the necessary security protocols to interact with that specific software system, such as Dynamics 365 or Salesforce.

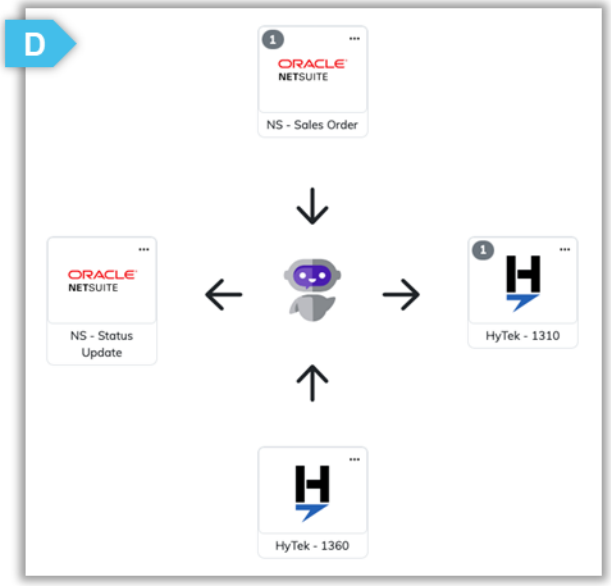
When an Endpoint (C) (connecting application) attempts to read or write data, it needs access to the connecting system, which it gets through an API. An API is the set of rules or protocols enabling the applications to communicate. When building Connectors, you'll provide the API credentials for each system that needs to connect to the Common Model. Every Connector has its own unique authentication process. You'll need to designate secure settings for each Connector to be able to authenticate and interact with that system – specifying selector authentication information such as BaseEndpoint, LoginEndpoint, etc.

⚠ **NOTE:**	Typically, secure settings data can be found in the Admin settings of the connecting system. Likewise, system-specific API protocols can be found on the company's website (e.g. Salesforce, ADP, etc.). Links to common third-party software selector authentication documentation can be found in Appendix A: Connector Catalog.

Build Data Hubs

After establishing the needed Connectors, you'll start building Data Hubs to organize how the data flows between systems. A Data Hub (D) is a logical grouping of Endpoints that form a single integration solution. For example, in your business there are events happening every day – a customer places a sales order, a new employee is hired, a product is shipped, etc. When an event happens, it's likely that you want that data recorded in multiple systems; so, you can create a Data Hub to include all of the systems that store employee information and assign Actions to ensure the data from one system is properly communicated to the next.

A Data Hub should be built like a module in a software solution, by clustering logical grouping to make it straightforward and easy to maintain. Each grouping is typically based on a specific entity (e.g. Customers) or a specific business process (e.g. Lead to Cash) Each Data Hub will connect two or more Endpoints to the Common Model. Endpoints can publish data to the hub (the Common Model is simply reading that data), subscribe to data from the hub (the Common Model is writing data to that Endpoint), or be bi-directional. Adding Endpoints to a Data Hub is easy; simply select the Endpoints you want the Common Model to connect to and then assign an Action – publish, subscribing, or both.

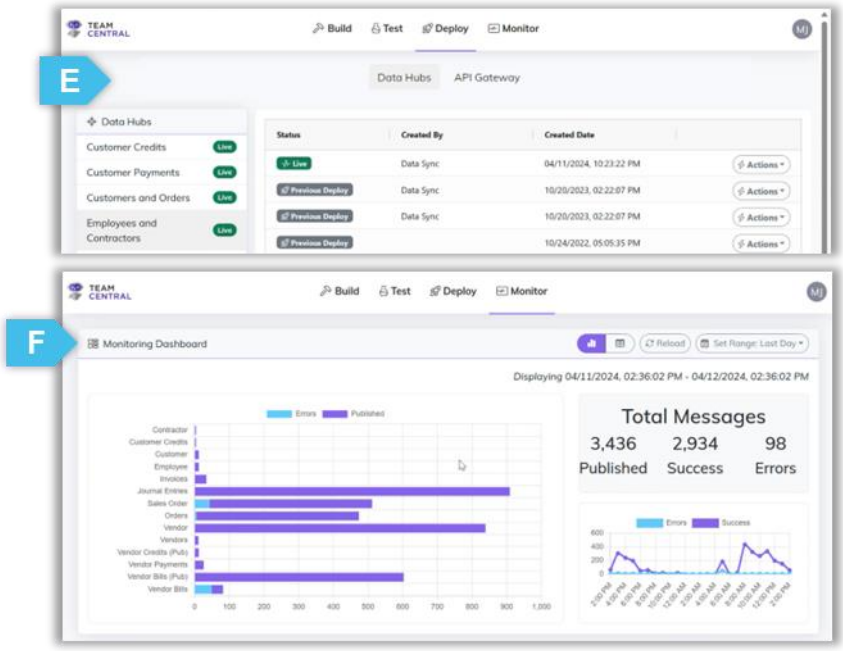


Best Practice: You should create as few or as many Data Hubs as necessary to connect your data. Within a Data Hub, the fewer Mappings you create, the easier it will be to manage the data. Each Data Hub has its own version and can be updated and deployed as needed; so, smaller hubs provide stronger logical groupings, with improved ability to modify and deploy the specific updates needed. Generally, it's better to start small and consolidate as needed.

Deploy and Monitor

When you decide that your configuration is ready to be pushed to a Live version, you can deploy (E) the Data Hub. Data Hubs have their own internal versioning system to help manage your work: (1) **Draft** – a work-in-progress, (2) **Live** – a version running in production, and (3) **Previous Deploy** – previous Live copies. Additionally, you can add comments, compare a draft version with the current live version or with a previous version, and view the code. Adding comments for each deployment provides context to other users and ensures all modifications are documented.

Once a Data Hub has been deployed, you can monitor (F) the integration and Actions completed within that Data Hub. The Dashboard tool enables you to view the health of the systems, including all messages. It also measures the number of errors, quantifying the number of successful data transfers versus errors occurring within each deployed Data Hub. The Message Search tool enables you to further investigate any errors on an individual, micro level, helping you to better evaluate any issues.



Build and Manage Schema Definitions

A Schema Definition is the whole of the data you are trying to define; while the components that make up that data are Schema Properties. For example, a Schema Definition would be *Customer*, while the *customer name* is a single Schema Property of that definition. You can define Schema Properties via three (3) methods – Simple, Foreign Key, and List. You can add multiple Properties, as well as multiple Property Types to a single Schema Definition.

Best Practice:

- It is strongly advised to develop standards for naming conventions and custom fields at an early stage, such as consistent case, where to use hyphens, etc.
- It is advised to use the Simple Schema Definition when available.

Schema Types

Simple Schema Definition

A simple Schema Definition is a single value to a single value transaction. It represents data that is both common and static, meaning it applies to all (or most) of your data systems, and it isn't likely to change within the systems you're synchronizing it to, such as a Customer Name. A list of common definitions is selectable from the simple Schema Definition entity list. When defining simple Schemas, your Schema Definition will need to align and link to a property defined by the Common Model (a Central Property).

Foreign Key Schema Property

Foreign Key Definitions are references to data (or values) that may be different from one system to the next, such as a customer record number in your sales system vs your billing system. These definitions are customer-built and customized to your specific needs.

List Schema Property

A List Schema is simply a pointer to another Schema and is used for hierarchy, or parent/child, customizations. For example, you may have a record that contains multiple components, such as an order has a list of line items, and each of those line items will have their own Schema to define components such as Quantity, Amount, Discounts, etc. Any parent/child data should be defined within a List Schema.

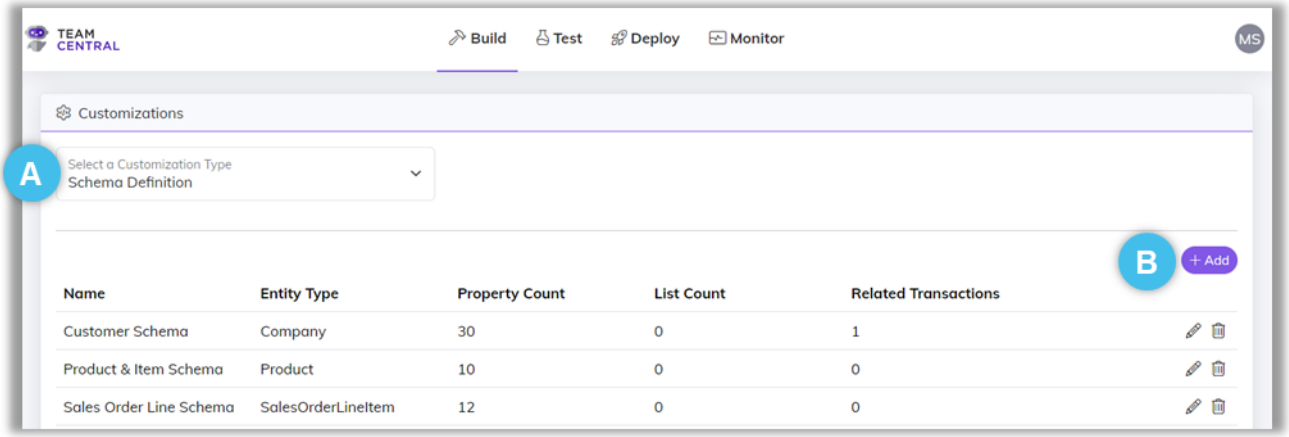
Transaction Types

Transaction types are Events that become part of the Schema, physically pairing a Publisher and a Subscriber. They identify the actions you'll want to take on this data, such as Save Customer or Delete Customer. Transaction Types tie directly to a Schema Definition, enabling users to select those actions when adding Endpoints.

NOTE:	Each Schema Definition must be assigned one or more Transaction Type(s). The Transaction Type(s) created during this process will become part of a selectable list when adding Endpoints to map your data.

Add a Schema Definition

1. Select **Build** > **Customizations** from the main menu.
2. Select **Schema Definition** (A) from the drop-down menu.
3. Select **+ Add** (B).



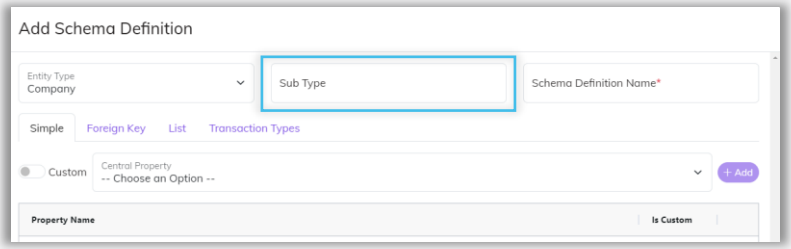
4. Select an **Entity Type** (C); and as needed, enter a **Sub-Type**.

NOTE: A Sub-Type can be defined for pre-determined Entity Types and will add a more precise attribute to that Schema. Sub-Types allow Schemas under one Entity Type to be cataloged into more specific logical groupings, and enables the Common Model to group these relationships in the database, even when the data may be housed in various parts of or entirely different systems.

For example, if you have several Schemas with the Entity Type “Company,” some may be classified as Sub-Type “Vendor,” while others may be classified as “Customer.” In this scenario, you would create two separate Schemas – one for your Vendor sync and one for your Customer sync. The Sub-Type discloses which one this particular relationship refers to – is the system looking for Customer data or Vendor data.

It is a freeform text entry field that will display when an applicable Entity Type has been selected, and cannot be edited once it is saved. This enables users to create more specific sub-groups of data within an Entity Type.

- The Sub-Type field will appear only for supported Entity Types.
- Each Sub-Type must be entered precisely as needed (e.g. casing, spacing, etc.). For example, if you enter Vendor vs vendors, you will end up with two (2) sub-types.
- When adding a new Entity Type to a Foreign Key, you can select to identify it as a previously created Sub-Type from the drop-down menu. See the Foreign Key section for additional information.



5. Enter a **Schema Definition Name** (D).

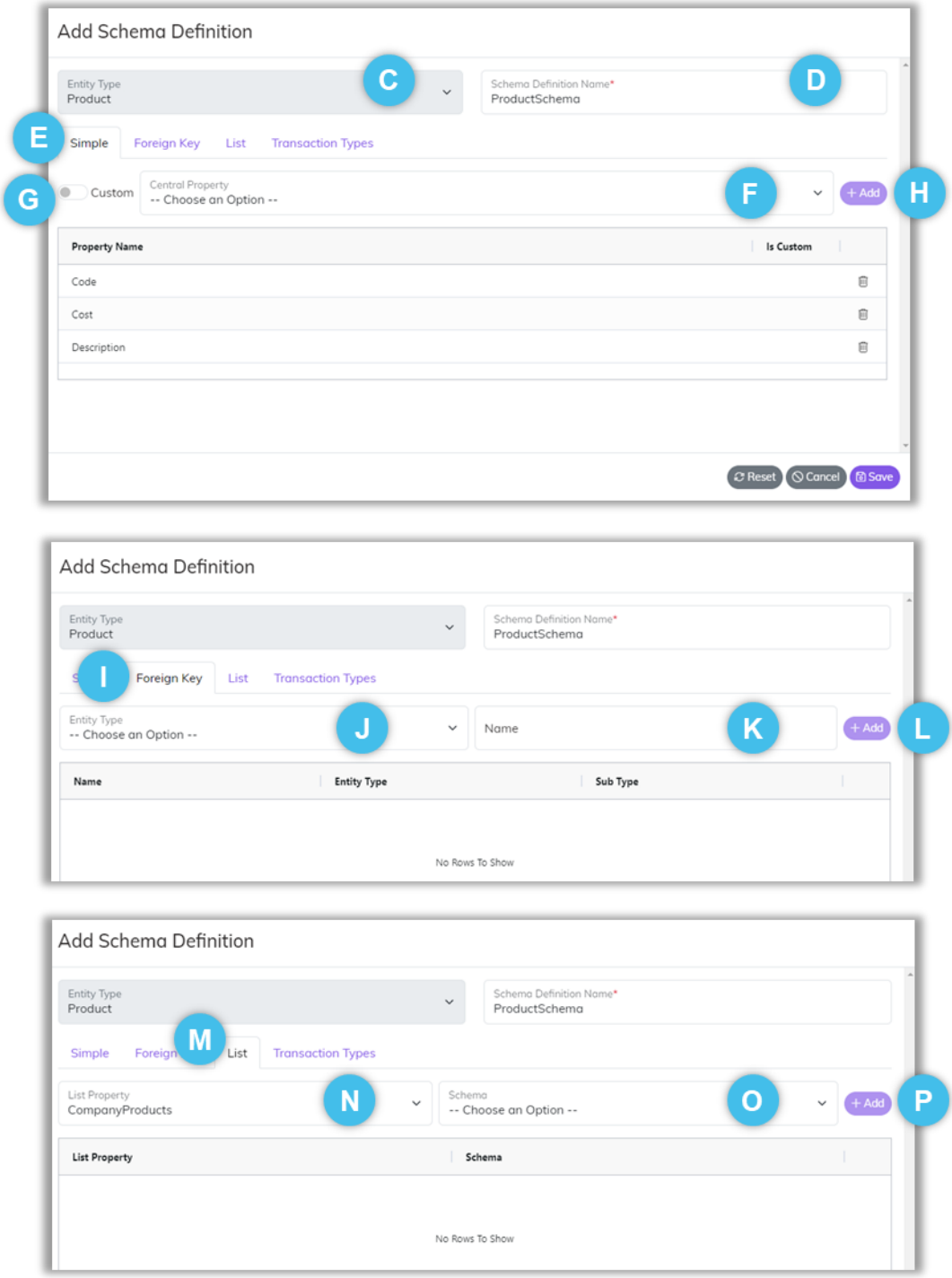
> **NOTE:** Do not add spacing to any field entries.

6. Then, use the below table for next steps:

> **NOTE:** You can add multiple Schema properties, of multiple types, to a single Schema Definition. However, to create a List Property, you must first define each list item as its own Schema Definition.

To:	Do this:
Add a Simple Schema	• Ensure the **Simple** tab (E) is selected. • Select a **Central Property** (F) from the drop-down menu; or, toggle **Custom** ON (G) to enter a custom Property into the text field. • Select **+Add** (H). The Central Property will be added to the table. • As needed, repeat this process to designate additional Properties to this Schema Definition. **Note:** As needed, see the example box below.
Add a Foreign Key	• Select the **Foreign Key** tab (I). • Select an **Entity Type** (J) from the drop-down menu; and if applicable, select a **Sub-Type** from the drop-down list. • Enter a **Name** (K) in the text field. • Select **+ Add** (L). • As needed, repeat this process to designate additional Properties to this Schema Definition. **Note:** The Sub-Type drop-down menu will only appear for applicable Entity Types.
Add a List Property	• Select the **List** tab (M). • Select a **List Property** (N) from the drop-down menu. • Select a **Schema** (O) from the drop-down menu. • Select **+ Add** (P). • As needed, repeat this process to designate additional Properties to this Schema Definition. **Note:** Only selectable options will be shown in both the List Property and Schema drop-down menus. The List Property menu will be filtered based on your Entity Type selection; and the Schema menu will be filtered based on your List Property selection.

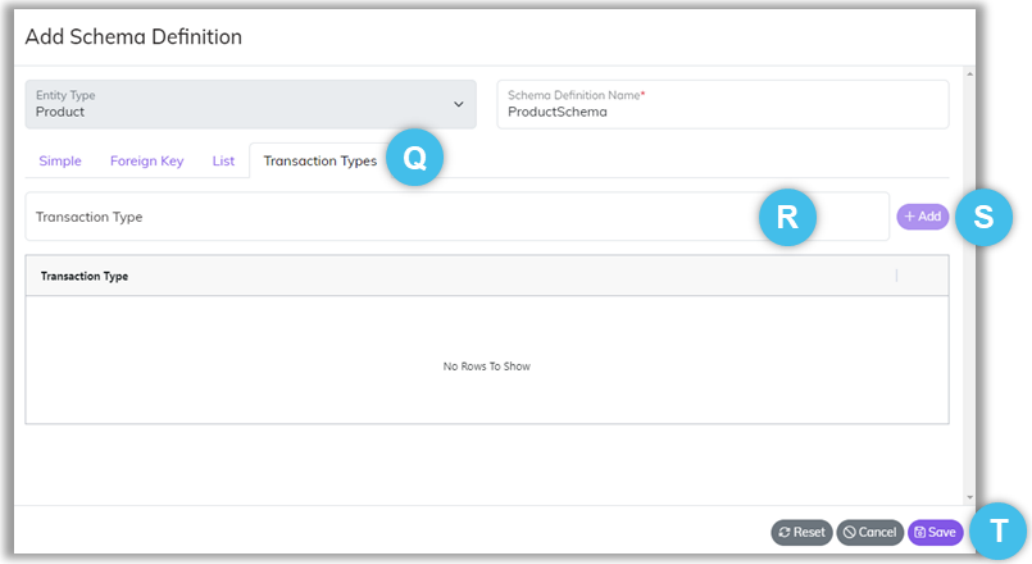
> **Example**: You may create a Schema Definition named *Customer Schema*, to define all pertinent company data; so, you may want to include Central Properties such as *Name*, *PrimaryBillingAddress1*, *PrimaryBillingAddress2*, *PrimaryBillingCity*, etc.


Add Schema Definition
Entity Type
Product
C
Schema Definition Name*
ProductSchema
D
E
Simple
Foreign Key
List
Transaction Types
G
Custom
Central Property
-- Choose an Option --
F
+ Add
H
Property Name
Is Custom
Code
Cost
Description
Reset
Cancel
Save
Add Schema Definition
Entity Type
Product
Schema Definition Name*
ProductSchema
I
Foreign Key
List
Transaction Types
Entity Type
-- Choose an Option --
J
Name
K
+ Add
L
Name
Entity Type
Sub Type
No Rows To Show
Add Schema Definition
Entity Type
Product
Schema Definition Name*
ProductSchema
M
Simple
List
Transaction Types
List Property
CompanyProducts
N
Schema
-- Choose an Option --
O
+ Add
P
List Property
Schema
No Rows To Show

7. Select the **Transaction Types** tab (Q).
8. Enter a **Transaction Type** (R) in the text field; then, select **+ Add** (S).
   The data will be added to the Transaction Type table below.

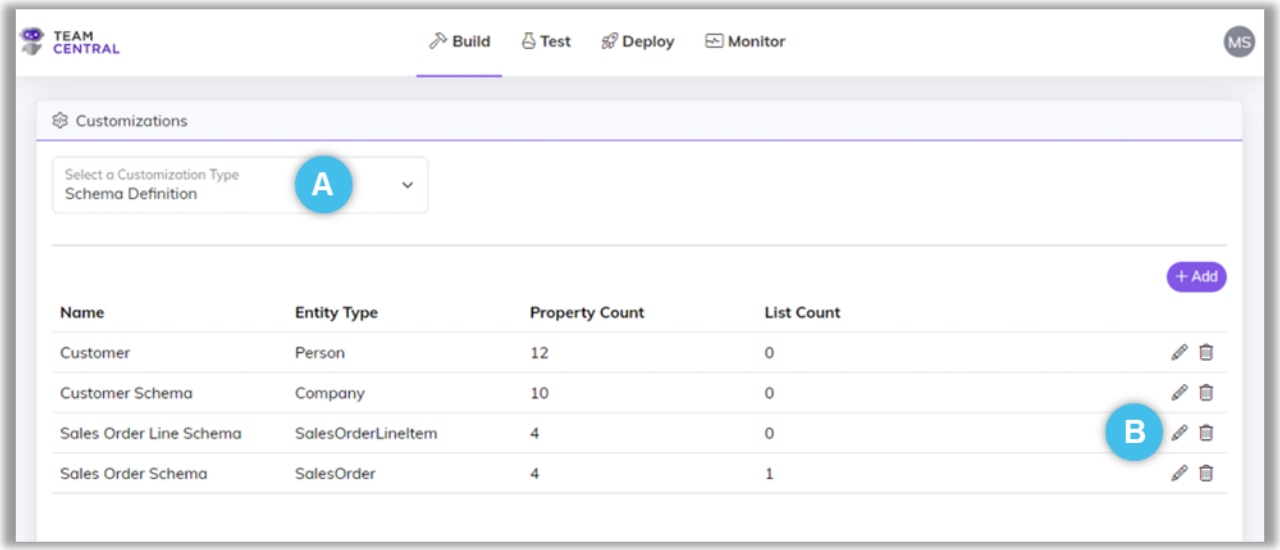
> ⚠ **NOTES:**
> - This is a freeform text field to describe the Event name, ultimately linking Publishers and Subscribers together. When messages get published under a certain Transaction Type, only subscribers to that Transaction Type will process it.
> - Multiple transaction types can be added to a single Schema Definition, and used for micro-level messaging, publishing the same data in several ways, or as various ways to target or isolate the integrations.
> - Users can build Schema Definitions and Transaction Types incrementally and in the order that feels most effective; and users can amend them as needed at any time. However, both the Schema Definitions and Transaction Type(s) must be completed prior to mapping Endpoints.

9. As needed, repeat this process to identify all necessary Transaction Types.
10. When finished, select **Save** (T).



Edit a Schema Definition

1. Select **Build** > **Customizations** from the main menu.
2. Select **Schema Definition** (A) from the drop-down menu.
3. Select the **edit** icon (B) in the appropriate Schema Definition row.

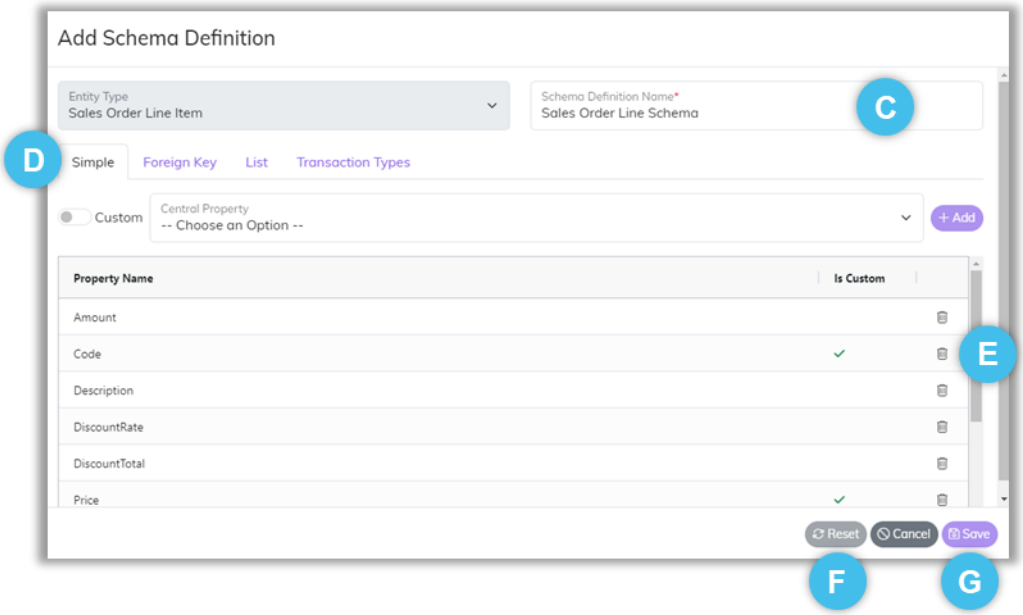


4. Modify the attributes as needed; use the below table for additional instructions.

To:	Do this:
Edit the Schema Definition Name	• Enter a new **Schema Definition Name** (C) in the text field.
Add a Simple Schema, Foreign Key, and/or List Definition	• See respective instructions in the *Build and Manage Schema Definitions* section of this manual.
Delete a Property or Transaction Type from the Schema Definition	• Select the appropriate tab – **Simple**, **Foreign Key**, **List**, or **Transaction Types** (D). • Select the **delete** icon (E) in the appropriate row.
Reset all edited attributes	• Select **Reset** (F) to discard all changes.

5. Select **Save** (G).

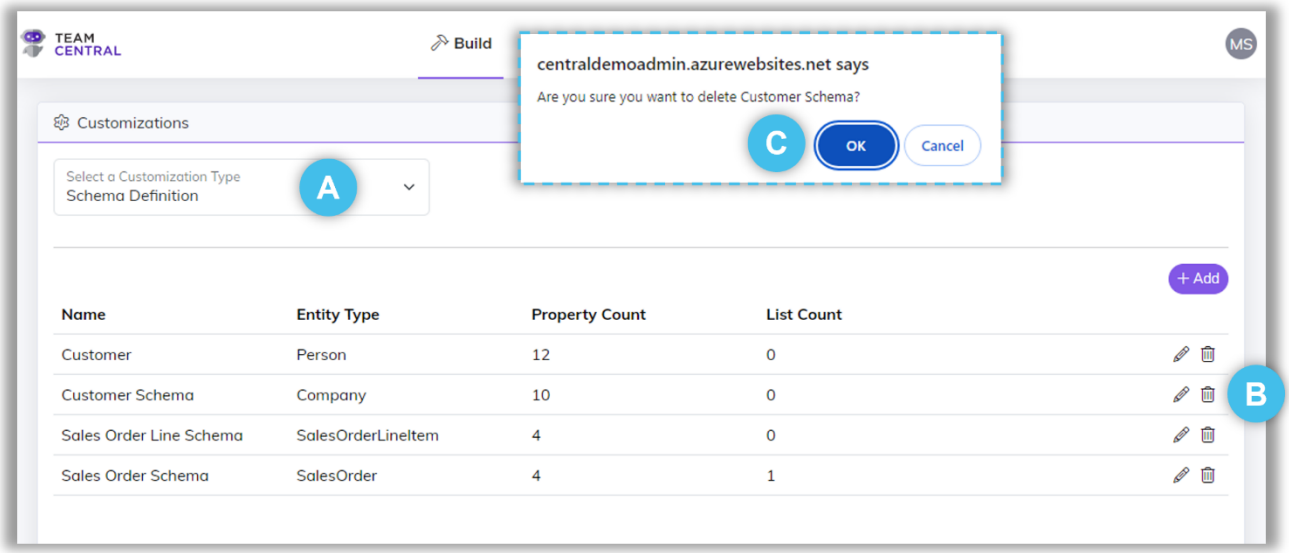
NOTE:	To discard all changes and exit edit mode, select **Cancel**.


Add Schema Definition
Entity Type
Sales Order Line Item
Schema Definition Name*
Sales Order Line Schema
C
D
Simple
Foreign Key
List
Transaction Types
Custom
Central Property
-- Choose an Option --
+ Add
Property Name
Is Custom
Amount
Code
Description
DiscountRate
DiscountTotal
Price
E
Reset
Cancel
Save
F
G

Delete a Schema Definition

1. Select **Build** > **Customizations** from the main menu.
2. Select **Schema Definition** (A) from the drop-down menu.
3. Select the **delete** icon (B) in the appropriate Schema Definition row.
4. Select **OK** (C) to confirm.

> **NOTE:** Deleting a Schema Definition cannot be undone; you must recreate it if needed.



Host Type Extension Schema Definition

NOTE:	Host Type Schema Definitions are intended for advanced users.

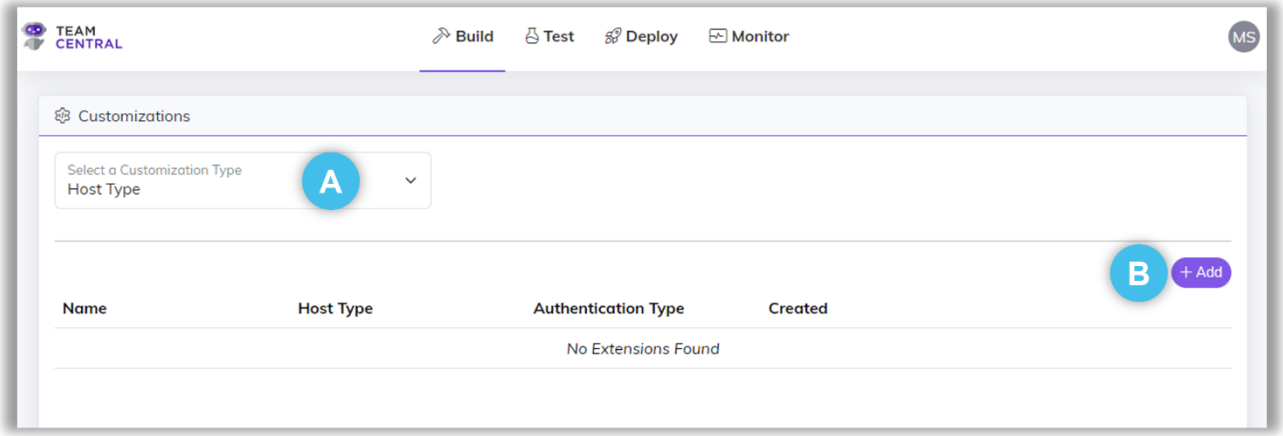
Advanced Tool: Host Type Extensions enable developers to take a pre-built Central Connector and extend it for your specific needs (i.e. build your own override). The *Host Type* references the location in the cloud where your customizations are hosted to run.

Central can do this in three (3) ways:

1. The connection runs embedded in the agent, which is usually done for network purposes. If Central needs to get data from within a company's network, the agent accesses the data sources directly.
2. The connection runs via a cloud-based Connector; Central runs on Microsoft Azure. To connect System A to System B, the Connector makes a call out to Central, which distributes the data via proxy method. (This is the typical method employed by Central.)
3. The connection runs via host type extensions that are built and customized by the developer, specific to your business needs. You build a company-specific space in the cloud that will host a Connector (e.g., Google Cloud, AWS, etc.); then override for the functionality you need to customize your software.

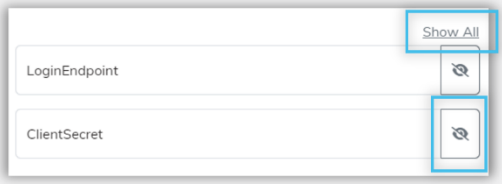
Add a Host Type Extension Schema Definition

1. Select **Build** > **Customizations** from the main menu.
2. Select **Host Type** (A) from the drop-down menu.
3. Select **+ Add** (B).

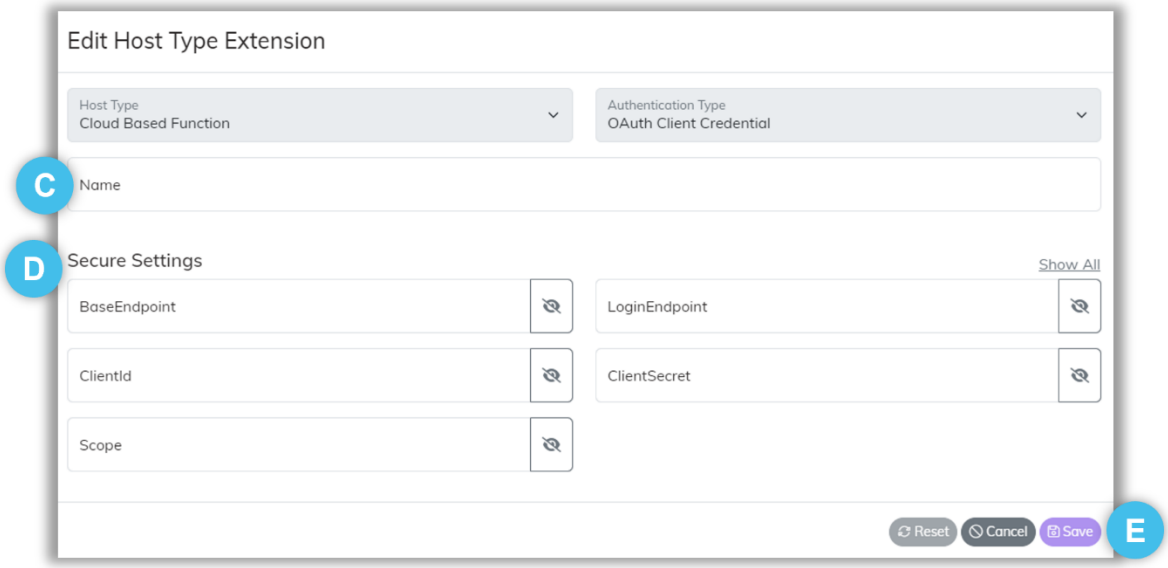


4. Enter a **Name** (C) in the text field.
5. Enter the appropriate **Secure Settings** (D).

NOTE:	To view the entered data, select **Hide All** or **View All** or toggle the **View/Hide** icon in that entry field.



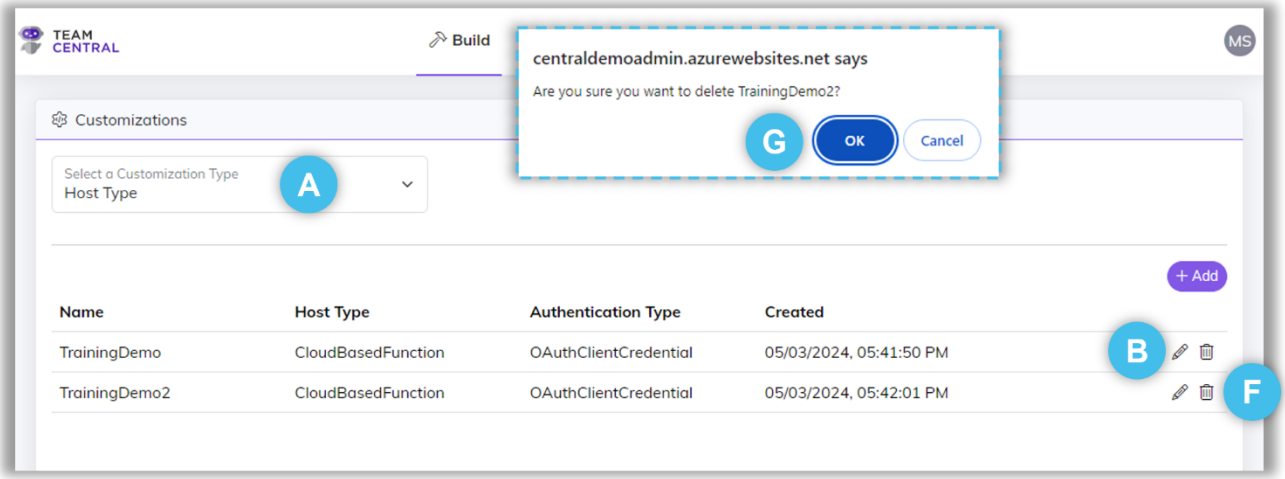
6. Select **Save** (E).

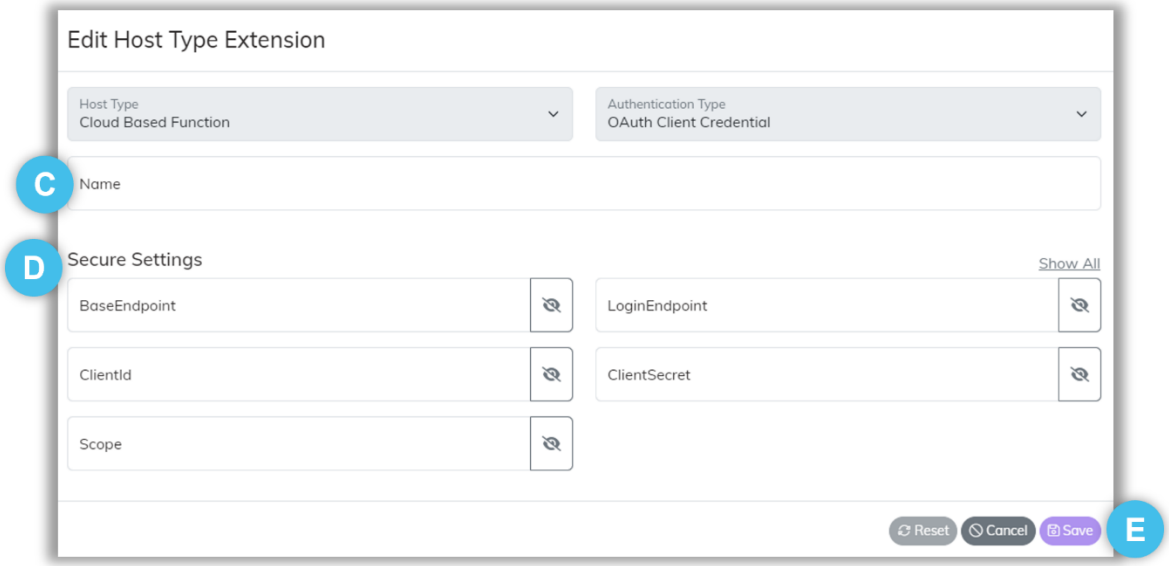


Edit or Delete a Host Type Extension Schema Definition

1. Select **Build** > **Customizations** from the main menu.
2. Select **Host Type** (A) from the drop-down menu.
3. Then, use the below table to determine the next step(s).

To:	Do This:
Edit a Host Type Extension	• Select the **edit** icon (B) in the appropriate row. • Modify the **Name** (C) and/or **Secure Settings** (D) as needed. • Select **Save** (E).
Delete a Host Type Extension	• Select the **delete** icon (F) in the appropriate row. • Select **OK** (G) to confirm.





Build and Manage Connectors

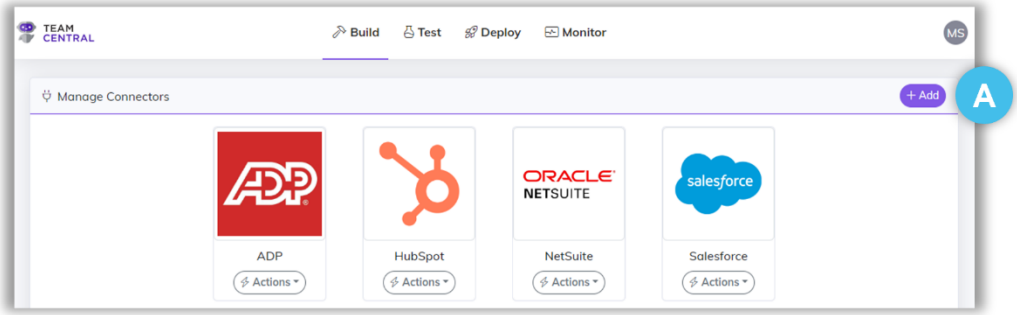
Connectors give Central the necessary security settings to interact with a specific software system, such as Dynamics 365 or Salesforce. You'll need to designate both a method and a protocol for each Connector. A method is simply how the Common Model interacts with the connected system, such as Get (read/query data) or Put (update data). A protocol is the data format that the API supports, such as JSON, SQL, or XML.

NOTE:	Since this is all pre-built in Central, there is no need to use Postman – the connectivity is already developed for you. Simply go into the Test Connector to write test requests against the Connector and check responses.

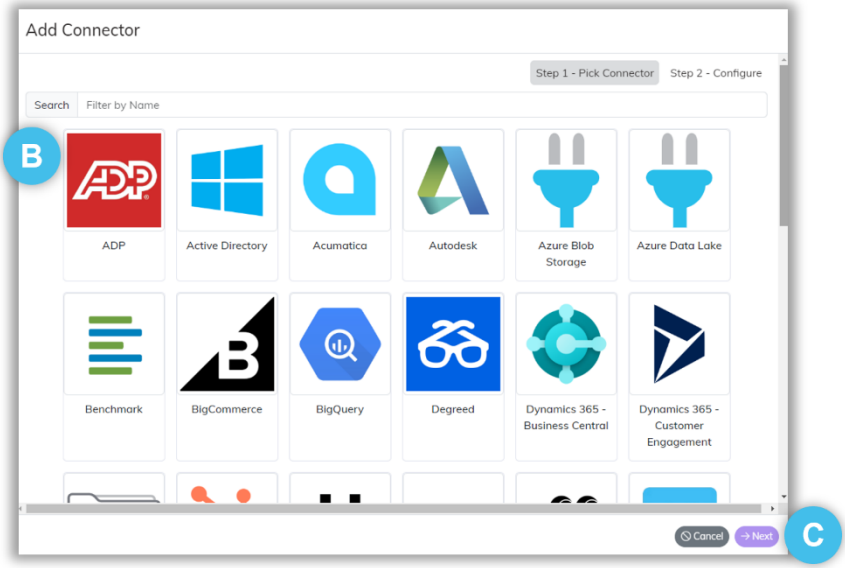
NOTE:	It is strongly advised to have the selector authentication information from each Vendor associated with the connections you plan to implement before starting this process. For links to common companies, see *Appendix A: Connector Catalog*.

Add a New Connector

1. Select **Build** > **Connectors** from the main menu.
2. Select **+ Add** (A).



3. Select a **Connector** (B).
4. Select **Next** (C).

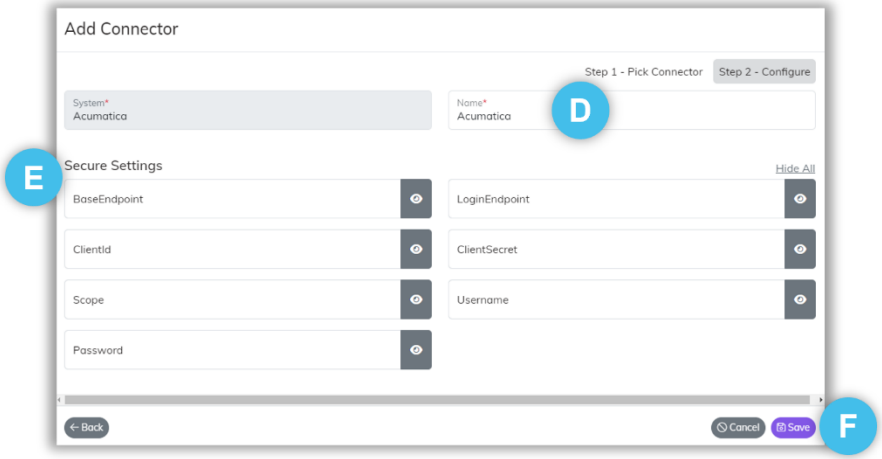


5. As needed, enter a new **Name** (D) for the Connector.
6. Enter the **Secure Settings** information (E).

NOTE:	The data entered for all Secure Setting fields must match your system's specific security details. This information can typically be found in the Admin settings of the system you are trying to connect.

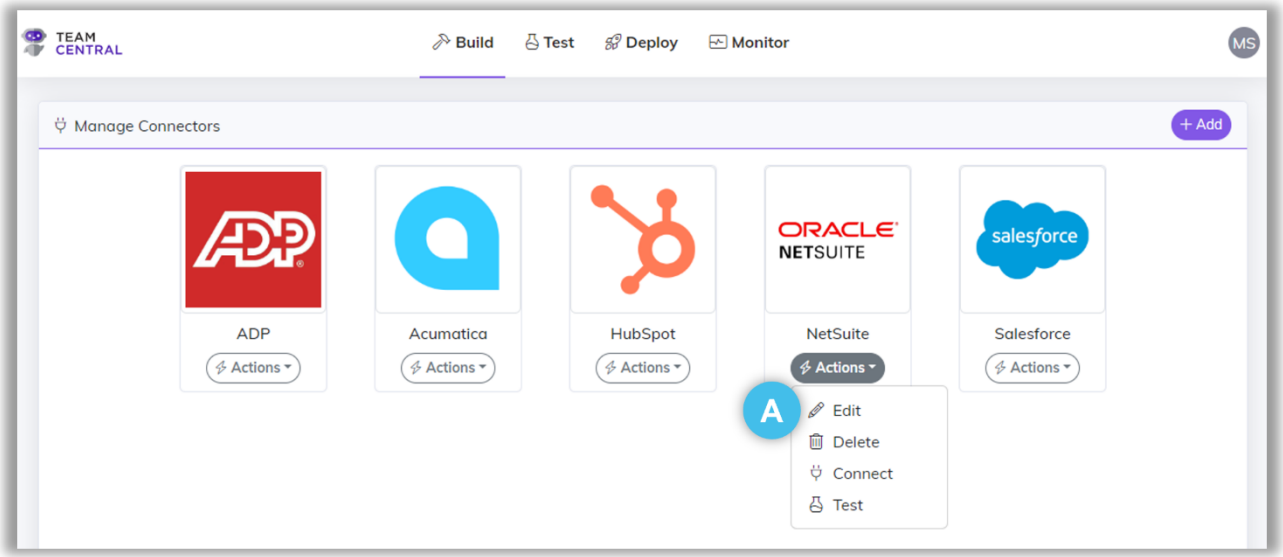
NOTE:	To view the entered data, toggle the **View**/**Hide** icon in each entry field, or select **Hide All** or **View All**. 

7. Select **Save** (F). The Connector will now appear on the Manage Connectors main screen.



Edit a Connector

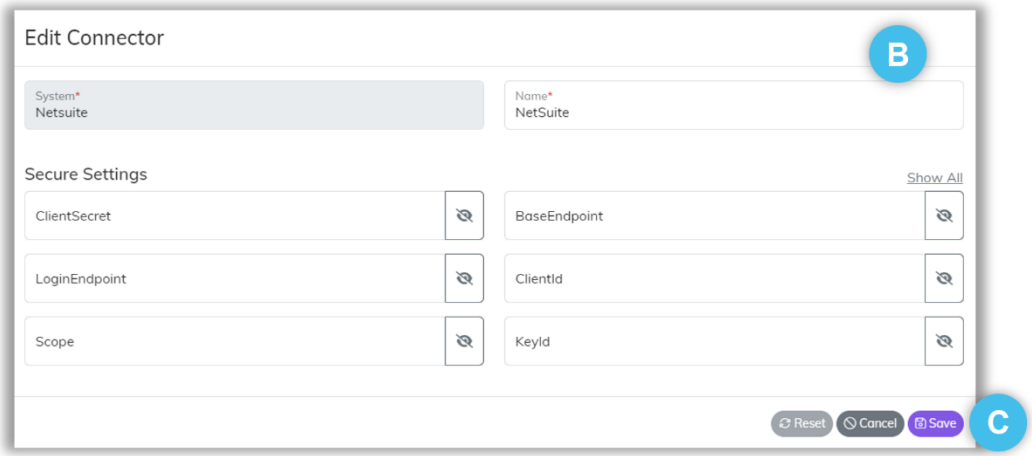
1. Select **Build** > **Connectors** from the main menu.
2. Select **Actions** > **Edit** (A) in the appropriate Connector box.



3. Modify the Connector **Name** and/or **Secure Settings** (B) as needed.

NOTE:	To discard all changes, select **Reset**. To discard all changes and exit edit mode, select **Cancel**.

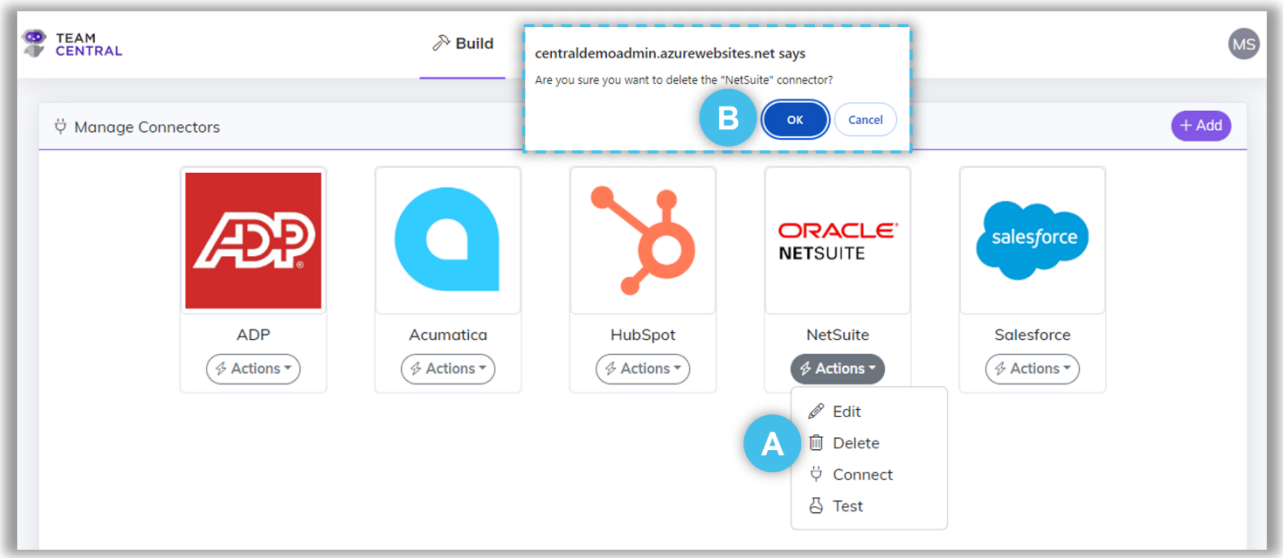
4. Select **Save** (C).



Delete a Connector

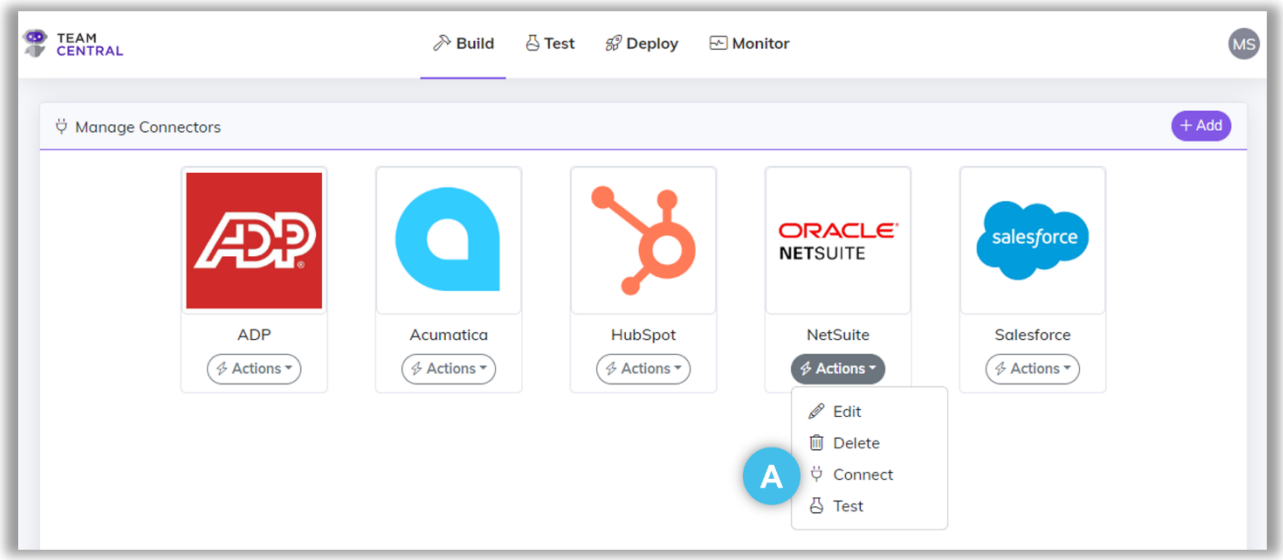
1. Select **Build** > **Connectors** from the main menu.
2. Select **Actions** > **Delete** (A) in the appropriate Connector box.
3. Select **OK** (B) to confirm the deletion.

⚠ NOTE:	Deleting a Connector cannot be undone; you must recreate it if needed.



Connect a Connector

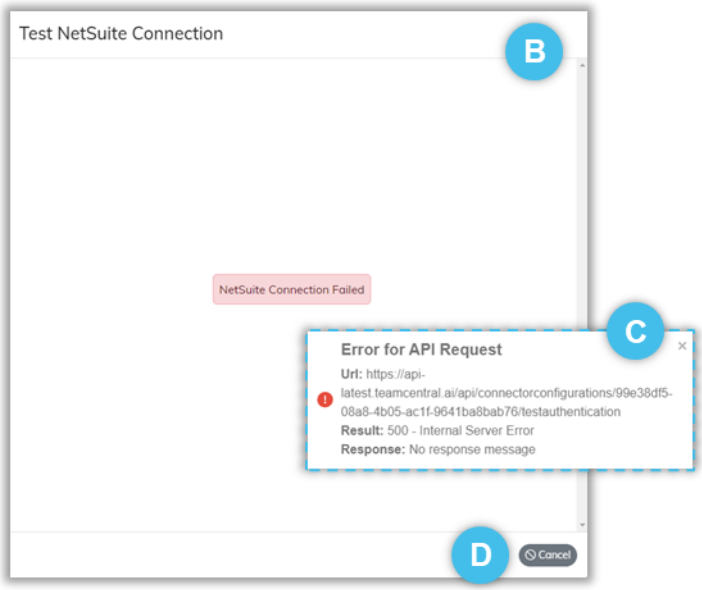
1. Select **Build** > **Connectors** from the main menu.
2. Select **Actions** > **Connect** (A) in the appropriate Connector box.



3. The Test Connection window (B) will appear while the system attempts to connect to ensure the secure settings are operational.

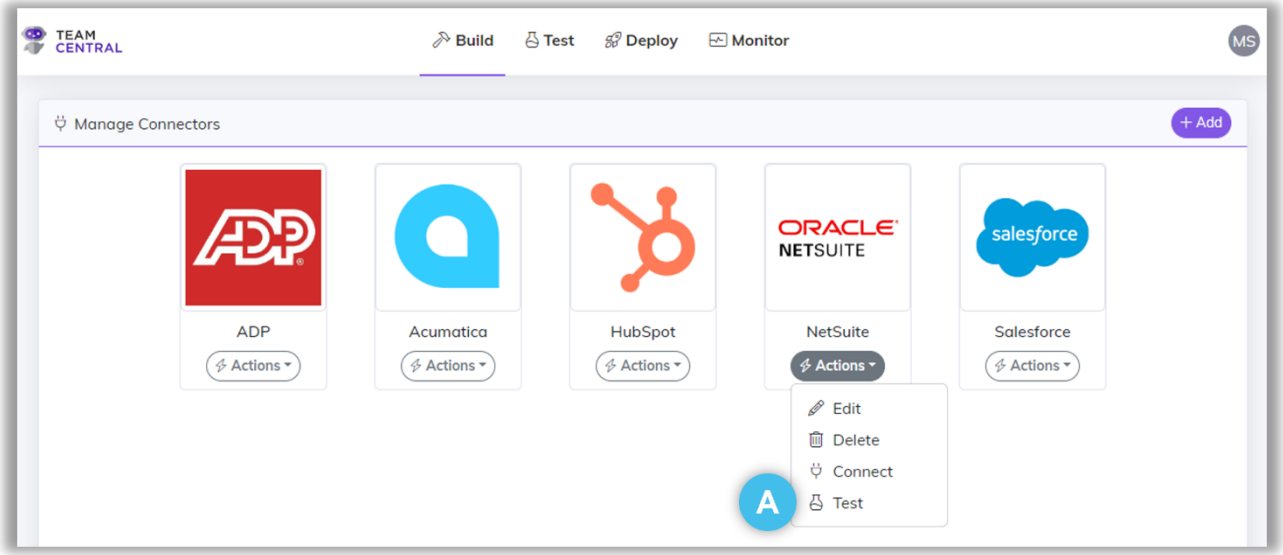
⚠ **NOTE:**	If an error occurs while trying to connect, a pop-up window (C) will identify why the connection failed. Select the **X** to close the error window.

4. When finished, select **Cancel** (D) to return to the Manage Connectors screen.

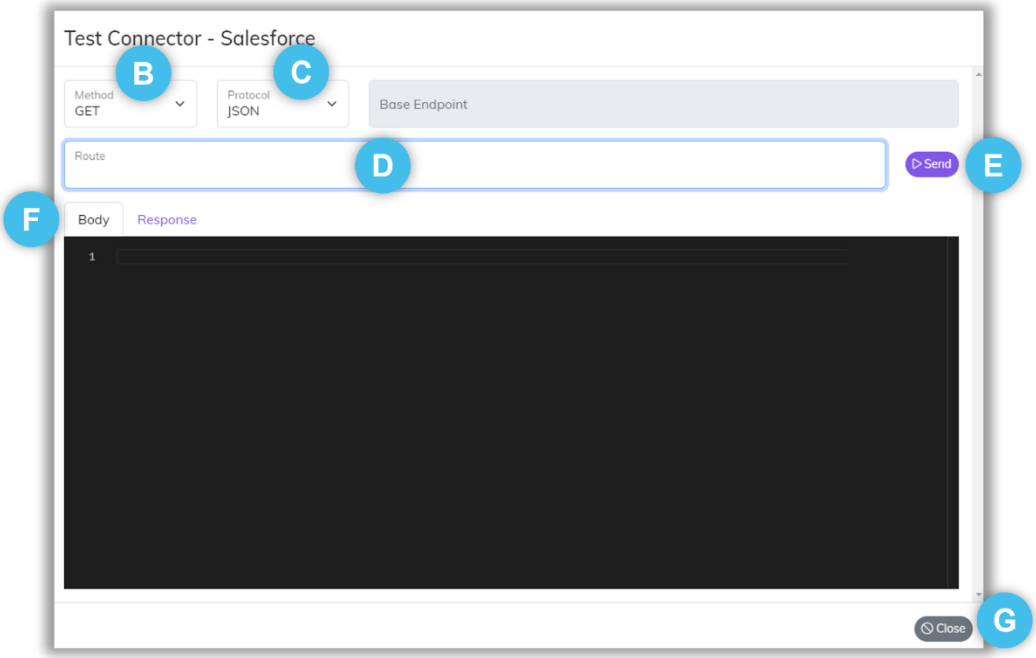


Test a Connector

1. Select **Build** > **Connectors** from the main menu.
2. Select **Actions** > **Test** (A) in the appropriate Connector box.

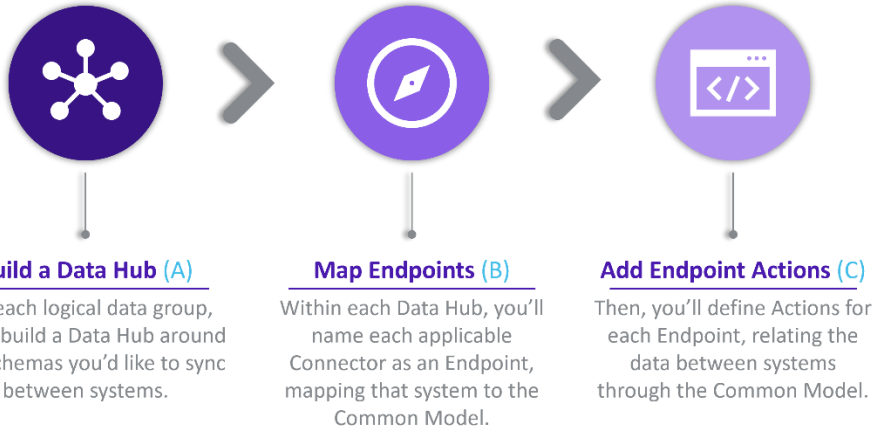


3. Select a **Method** (B).
4. Select a **Protocol** (C).
5. Enter **Route** data (D).
6. Select **Send** (E).
7. The connection test will show on the **Body** tab; the outcome will show on the **Response** tab (F).
8. Select **Close** (G) to exit the test window.



Build and Manage Data Hubs

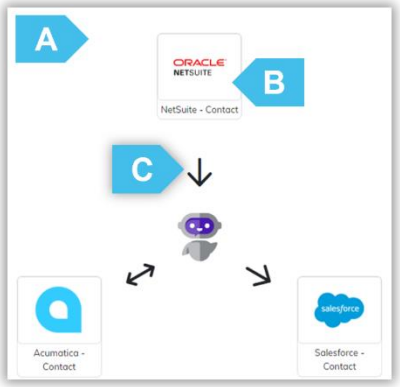
Building a Data Hub consists of creating a new Data Hub, adding Endpoints, configuring the Action(s) that Endpoint can execute, and then Mapping Endpoints Actions to the Common Model. This is the process that will tie all of the data together using your previously defined Schema Definitions and configured Connectors.



Build a Data Hub (A)
For each logical data group, you'll build a Data Hub around the Schemas you'd like to sync between systems.

Map Endpoints (B)
Within each Data Hub, you'll name each applicable Connector as an Endpoint, mapping that system to the Common Model.

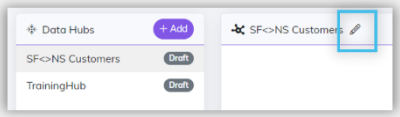
Add Endpoint Actions (C)
Then, you'll define Actions for each Endpoint, relating the data between systems through the Common Model.



Build a Data Hub

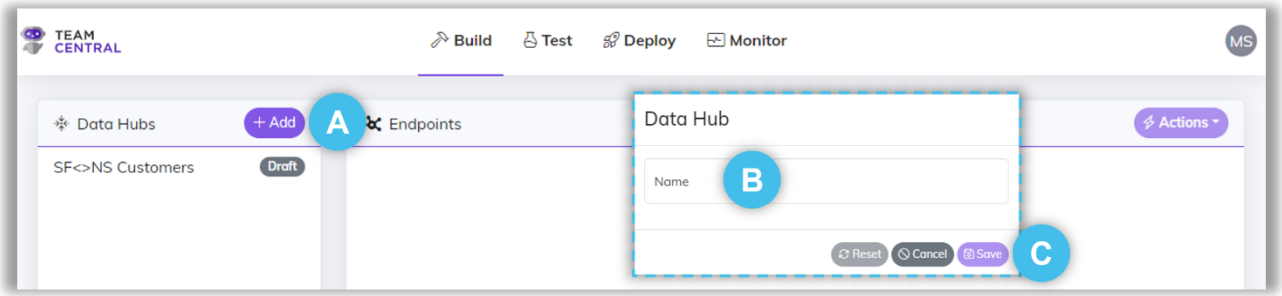
1. Select **Build** > **Data Hubs** from the main menu.
2. Select **+ Add** (A).
3. Enter a **Name** (B) for the new Data Hub.

NOTE:	Once a Data Hub is created, users can select the **edit** icon to rename it, if needed.



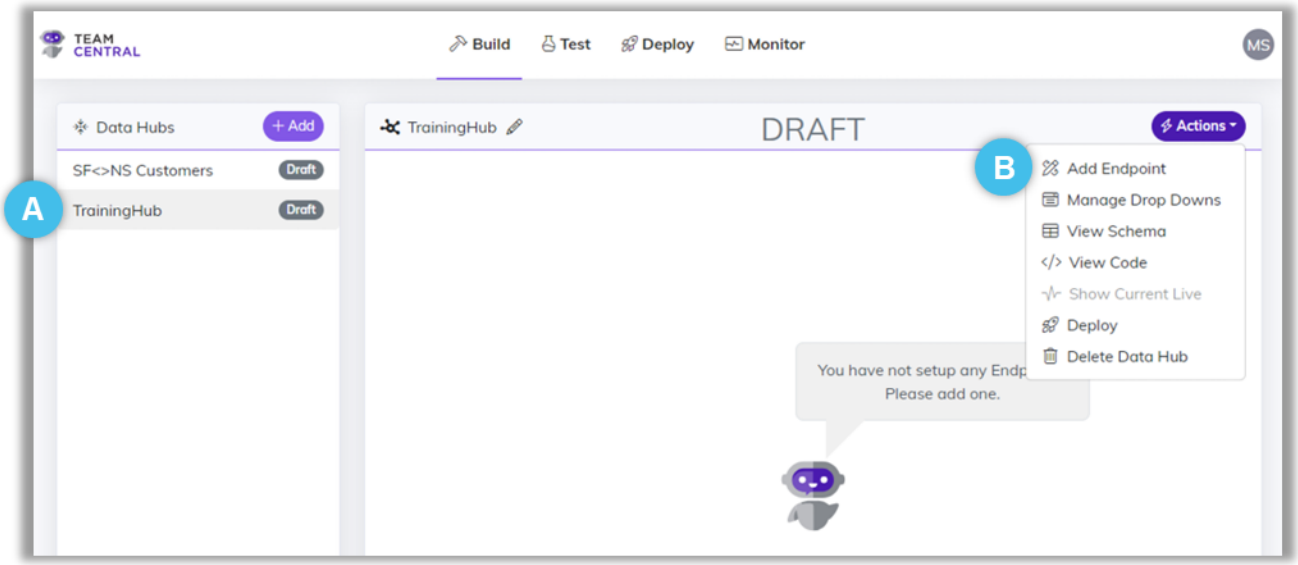
4. Select **Save** (C).

NOTE:	When a new Data Hub is created, it will be saved as a Draft version until deployed.

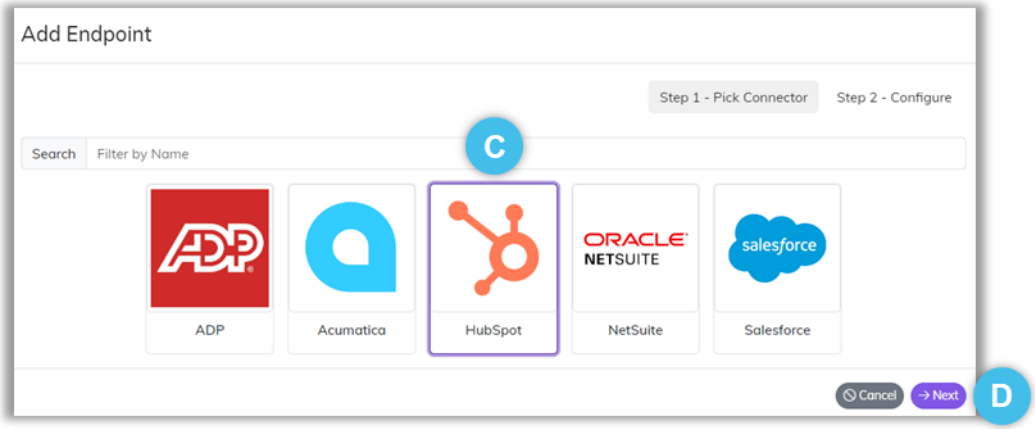


Add Endpoints to a Data Hub

1. Select the appropriate **Data Hub** (A) in the left column.
2. Then, select **Actions** > **Add Endpoint** (B).



3. Select the appropriate **Connector** (C).
4. Select **Next** (D).



5. Configure the **Action** by selecting options from the drop-down menus (E).
   For additional information about each selection, use the tables below.

Add Endpoint Selections

Field	Options
Template Options	• Select a **Template** from the drop-down menu. **Note:** Templates are pre-configured Endpoint definitions for a specific Connector, which gives a starting point for configuring Actions and Maps. ⚠ This feature is currently only applicable for specific functionality within certain Connectors; contact TeamCentral support for additional help.

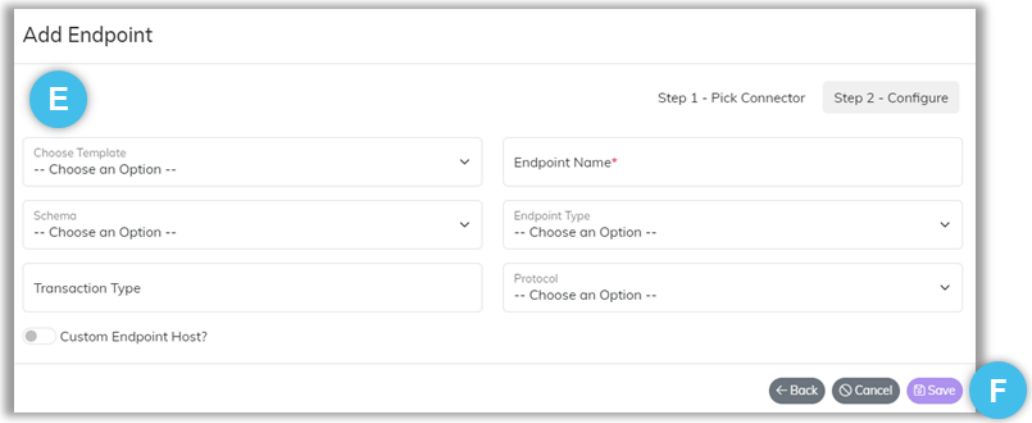
Schema Options	• Select the Schema that you want tied to the Endpoint.
Endpoint Name	• This field will be entered automatically based on the selected Endpoint, and is simply a displayed label in the design view
Endpoint Type	• Select the type of Endpoint. • Select from: Both (i.e. publish & subscribe), Both with Webhooks, Data Provider, Delete Publisher, Publish, Publish with Webhooks, Subscribe. **Notes:** • A single direction Endpoint will create either a save action or a read action (e.g. publisher will create only a read action, while a subscriber creates only the save action). • A bi-directional Endpoint will create both a save and read action. • When a save action is created, an interaction to the Key will be automatically created • A Webhook is a specific type of publishing Endpoint, where data is pushed from the Connector to Central; only certain vendors support this capability. • Data Provider is used when creating API Gateway Endpoints and should not be selected when working within a Data Hub.
⚠ Transaction Type	• Select a Transaction Type from the drop-down list. (The displayed options were created and tied to each Schema Definition.) • It is common to have multiple Transaction Types in the same Data Hub; however, it is best practice to keep all of the Endpoints that are publishing and subscribing to the same Transaction Type grouped within the same Data Hub. • See *Build and Manage Schema Definitions* section for additional information about Transaction Types.
Protocol Options	• Typical protocol could options include JSON, XML, Character delimited file, and SQL.
Custom Endpoint Host	• If you created a custom Host Type Extension, toggle the **Custom Endpoint Host** ON. • Select a **Host Type Extension** from the drop-down menu.

Endpoint Selection Option Descriptions

Selection	Description
Endpoint Type Options	
Batch Publish	Publishes a group of Transactions together.
Batch Subscriber	Receives a group of Transactions to be processed together.
Both	Publish and subscribe the Transaction.

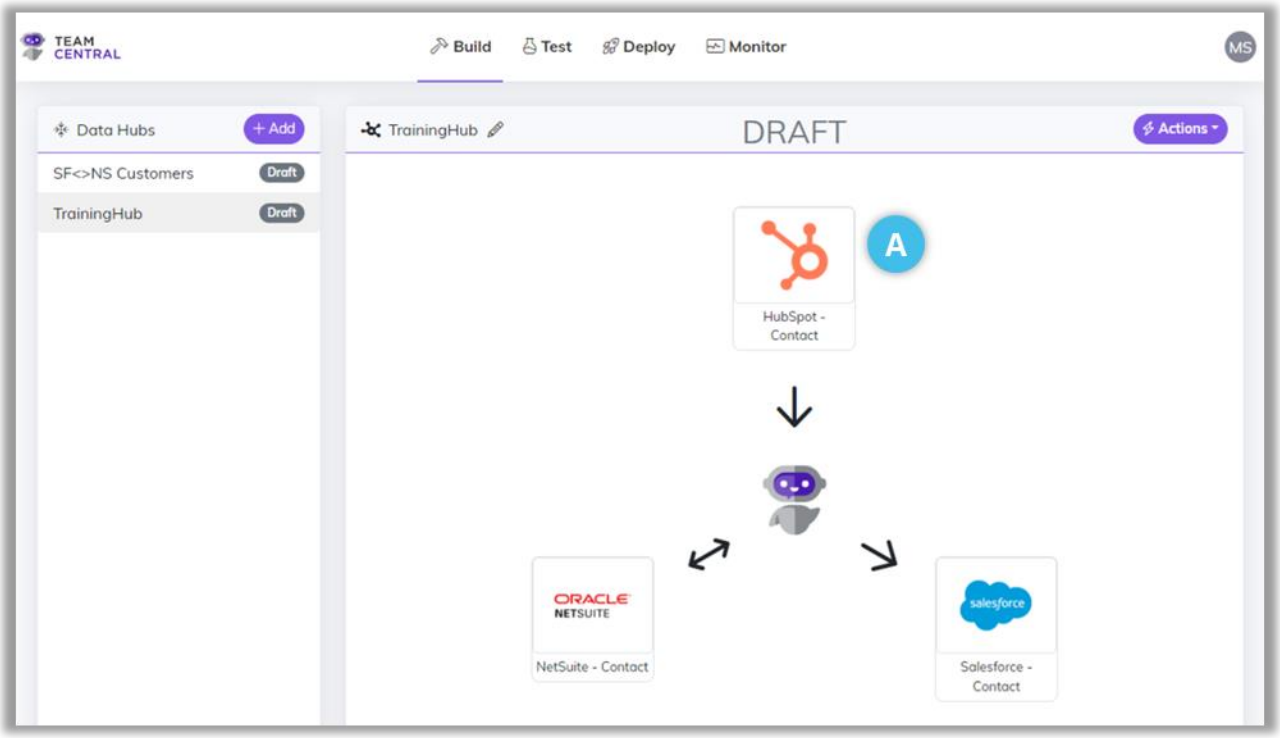
Both with Webhooks	A bi-directional Endpoint, where the Publisher is implemented using Webhooks. (See notes for Endpoint Type in the *Add Endpoint Selections* table above for additional details on Webhooks.)
Data Provider	A read-only Endpoint used in the API Gateway; it is not intended to be used inside of a Data Hub.
Delete Publisher	A Publishing Endpoint that is specifically used to publish records to be deleted or inactivated across systems.
Publish	Triggers the Publishing of data to be consumed by other systems that are listening for a particular Transaction Type.
Publish with Webhooks	A Publisher that is implemented with Webhook capability. (See notes for Endpoint Type in the *Add Endpoint Selections* table above for additional details on Webhooks.)
Subscribe	Triggers the Subscribing of data by systems listening for a particular Entity Type and Transaction Type.
Endpoint Protocols	
Delimited File	Refers to any type of file that usually participates in a Batch Publisher or Subscriber. In the USA, typical delimiters are commas.
JSON	A typical format for all API based integrations.
Other	Can be used to trigger the default key/value pair mapping UI, e.g. Active Directory.
SQL	A format for any data going into a Relational Databases, such as a SQL Server.
XML	This is a legacy format used by older API integrations.

6. Select **Save** (F).
7. Repeat the previous steps to add as many Endpoints as needed to this Data Hub.

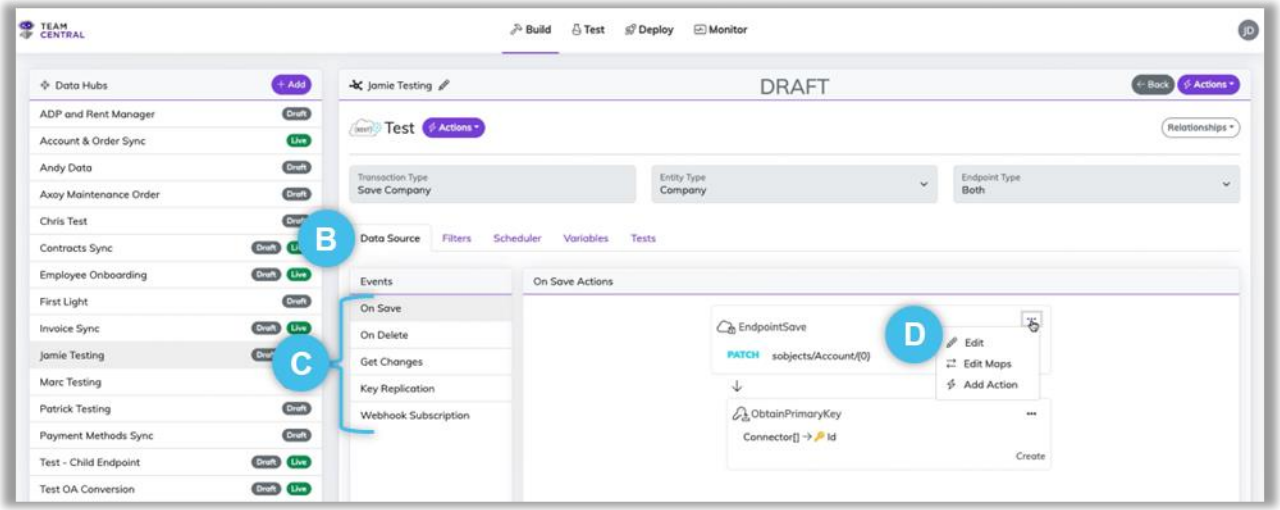


Configure Endpoints

1. Ensure the appropriate **Data Hub** is selected; then, select an **Endpoint** (A) to configure.



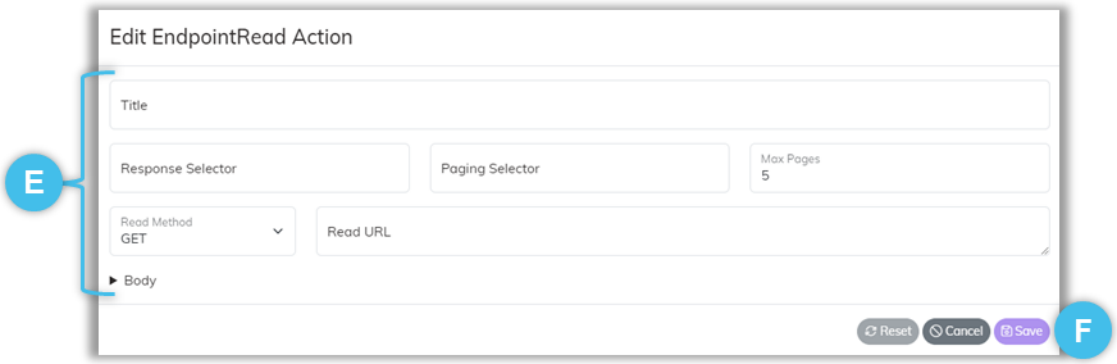
2. Ensure the **Data Source** tab (B) is selected.
3. Select the appropriate Endpoint action **Event** tab (C) from the left column.
4. Select the **menu** icon > **Edit** (D) to configure the Endpoint Action.



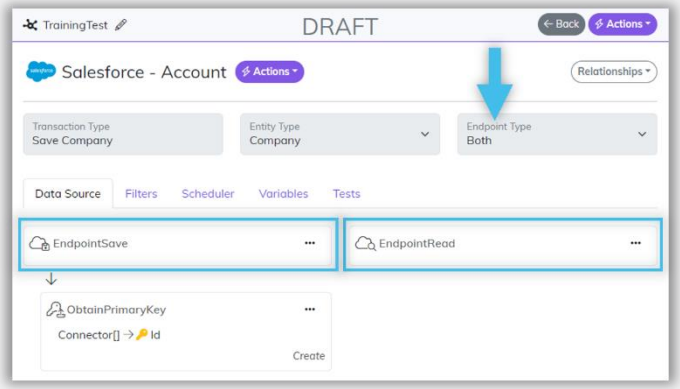
5. Configure the **Endpoint Action** (E).

NOTE:	The data required to configure an Endpoint Action is dependent on the action assigned (i.e. API Read, SQL Save, File Read, etc.). See the *Endpoint Actions Inputs* table below for detailed descriptions on each protocol.

6. Then, select **Save** (F).
7. Repeat the previous steps until all Endpoints Actions have been configured.



NOTES:	• You will need to configure each Endpoint Action created (e.g. if Both was selected as the Endpoint Type, then you'll need to configure the Endpoint Action for both the On Save and Get Changes events). • When a save action is created, the Common Model will automatically create an interaction to the Key. Users can create as many interactions as needed associated with the save action. • The **Read URL** field is critical for successful Endpoint Mapping, which you will construct next. When Mapping Endpoints, it is critical to ensure the data from a particular location within the payload gets into the Common Model. For additional information on common protocols (e.g. JSONPath, XMLPath, etc.) see *Appendix A: Connector Catalog*.



Endpoint Action Inputs

Input	Description

API Read	• **Title:** A freeform label you give to the Action.
	• **Response Selector:** JSON Path or Xpath query that points to where the data is on the response.
	• **Paging Selector:** JSON Path or Xpath query indicating where in the response to pull the URL of the next page.
	• **Max Pages:** Limits the total number of pages that the action can return.
	• **Read Method:** The Http Verb required by the API.
	• **Read URL:** the Http Address required by the API. This value is concatenated to the Base URL on the Connector.
	  ○ See *Parameterizing* section below for additional information.
	• **Read Body:** This field is not required, but in some cases may be useful, particularly for more advanced searching APIs. A POST is used in order to query, and in those cases an Http Body is used in the request.
	  ○ See *Parameterizing* section below for additional information.
	• Examples may include:
	  ○ SuiteQL
	  ○ SOQL
	  ○ OData
	  ○ Calling Select Stored Procedures – Date Last Polled and Parent-Child

API Save	• **Title:** A freeform label you give to the Action. • **Create Method:** The Http Verb required by the API to insert a new record. The default verb for a Create is a POST • **Create URL:** The Http Address required by the API to insert a new record. • **Update Method:** The Http Verb required by the API to update an existing record. The default method for an Update is a PATCH, but in some cases a PUT or POST can be used as well. • **Update URL:** The Http Address required by the API to update existing data. The URL for an Update is usually parameterized with the ID of the record being updated.
API Delete	• **Title:** A freeform label you give to the Action. • **Delete Method:** The Http Verb required by the API to delete a record. The default method for a Delete is DELETE • **Delete URL:** The Http Address required by the API to delete a record. The URL for a Delete is usually parameterized with the ID of the record of the record being deleted.

SQL Read	• **Title:** A freeform label you give to the Action.
	• **Read Command Type:** The SQL Command can be a Stored Procedure or Dynamic SQL Text.
	• **Read Command:** This should be set to the name of the Stored Procedure.
	  ○ On a Stored Procedure that can pull child items (e.g. Lines on an Order) it MUST have an input parameter called @ParentId.
	  ○ If a Stored Procedure that can pull data needs a date filter/parameter, it should be called @DateLastPolled.

SQL Save	• **Title:** A freeform label you give to the Action. • **Create Command Type:** The SQL Command can be a Stored Procedure or Dynamic SQL Text. • **Create Command:** If the SQL Command is Stored Procedure, then the command is the name of the Stored Procedure. • **Update Command Type:** The SQL Command can be a Stored Procedure or Dynamic SQL Text. • **Update Command:** If the SQL Command is Stored Procedure, then the command is the name of the Stored Procedure. If the SQL Command is Dynamic SQL Text, then the command is the parameterized query of the database. ○
SQL Delete	• **Title:** A freeform label you give to the Action. • **Delete Command Type:** The SQL Command can be a Stored Procedure or Dynamic SQL Text. • **Delete Command:** If the SQL Command is Stored Procedure, then the command is the name of the Stored Procedure. ○
File Read, Save	• **Title:** A freeform label you give to the Action. • **Location:** The folder or container name. • **File Name:** The name of the file. • **Delimiter:** The character that is used to parse the file; the default value is a comma.

Parameterizing

Parameterizing enables data to be dynamically passed into an action as a way of telling the mapping engine that some data needs to be injected, and potentially formatted, into the URL or Body of the API Request or a File Name. Parameterizing enables users to take the ID off the inbound message and append it to the Update message.

Instances where parameterizing can be used might include: to Update and Delete a record, Query date last polled from URL, Query date last polled from Body, and/or Add Variables to a Body.

Parameter	Description & Examples
System Parameters	Data that comes internally to the system. • SYS:CurrentDateTime will inject the current date/time.
Schema Parameters	Data that comes off another piece of data or message that is flowing through the system. • CD:{Name} will find custom data with matching name and inject that value, e.g {CD:CustomFieldA}. • PROP:{Name} will find a property on the model with a matching name and inject that value, e.g. {PROP:FirstName}. • PK:{Name} will find a Primary Key with the given Name for the processing system and inject that value, e.g. {PK:Id}. • FK:{Name} will find a Foreign Key with the given Name for the processing System and inject that value, e.g. {FK:CustomerId}.
Variable Parameters	Enables users to set any static value and inject it into an action, such as a specific value within a file name or parameter within a URL. • VAR:{Name} will find a Variable with the given Name and inject that value, e.g. {VAR:SomeVariable}.

NOTE:	Optionally, formatting can be applied to the injected value by adding an additional :{format} to the placeholder, e.g. {VAR:CurrentDate:yyyy-MM-dd}.

Key Scraping Action

When a record is created inside of a system, the Central engine can retrieve the new key generated for that record. Central will then save that key inside its database to be used for any subsequent call to update that record. This is referred to as *indexing the record*. Keys that have been scraped and stored inside of an index can also be used as a cross-reference on Foreign Key Maps.

Input	Description
Key Scraping	• **Title:** A freeform label you give to the Action. • **Key Name:** The name that the key is given inside of the index. The default value is ID; however, you are able to give the Key a more specific name. • **Key Source:** Connectors (directing it to look in the save response for the key) and Response Headers (directing it to look in the response headers for the key). • **Connector Property:** If the source is a Connector, it will be a path to the property of the key, e.g. “ID.” If the source is Response Headers, it will be a path to the specific header that holds the key, e.g. "..Location[0]"

⚠ **NOTE:**	Any Source other than Connector must be done using Code View. If you are unable to link a Key to a Connector, please contact TeamCentral Support.

Endpoint Relationships

Additional Endpoints can be added as a Child Endpoint or as a Dependent Publisher as well.

A Child Endpoint can be added to a Parent Endpoint to establish a hierarchy of data, such as a sales order that has line items or a project that includes multiple resources. In some cases, you can save and query child records with the parent; and in those instances, you can nest the maps of the children inside the parent (see *Map Endpoint Actions* section for *Looping Lists*). However, in other cases the data retrieval or save must happen separately per the API specifications, which is when a Child Endpoints can be used.

A Child Endpoint enables data to be pulled from different locations within the same system, using the same Endpoint. For example, if you establish an endpoint to pull sales order data, but the header data and the line-item data are stored in different locations, you can set-up one or more Child Endpoints so that all of the needed data is pulled within a single endpoint. When Central goes to either retrieve or save that order's data, it'll also include any Child Endpoints to ensure data is read/written appropriately.

A Child Endpoint functions the same way that the Parent Endpoint functions. In a read action, the Primary Key of the parent is "passed" into the Child Endpoint's configuration to query all of the children (see *Map Endpoint Actions* section for *Parameterization*). In a save action, the parent will loop and call that save action for each Child Endpoint, for each child entity that was received on the message payload. Each Child Endpoint must be set-up and mapped using the same steps as the Parent Endpoint.

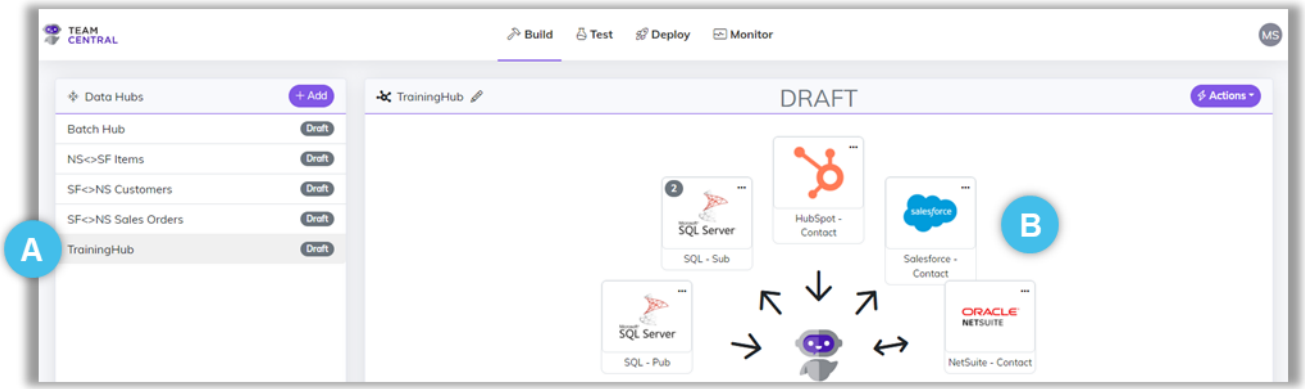
By contrast, Dependent Publishers are used when the synchronization between two different types of entities is contingent on one another. For example, a synchronizing sales order is typically related to a customer that will eventually be sent a bill. So, you likely cannot create the sales order until the associated customer has been synchronized first. In this scenario, you would have an Endpoint as a publisher for the customer data, and that Endpoint would have a Dependent Publisher for the sales order to ensure that Central does not attempt to synchronize a sales order until the customer dependency has been synchronized first.

Additionally, in some cases you may want to publish a message for data that has been deleted, so that other systems can also delete or deactivate that data. You can set up a Delete Publisher in a similar way, but with a couple of important notes:

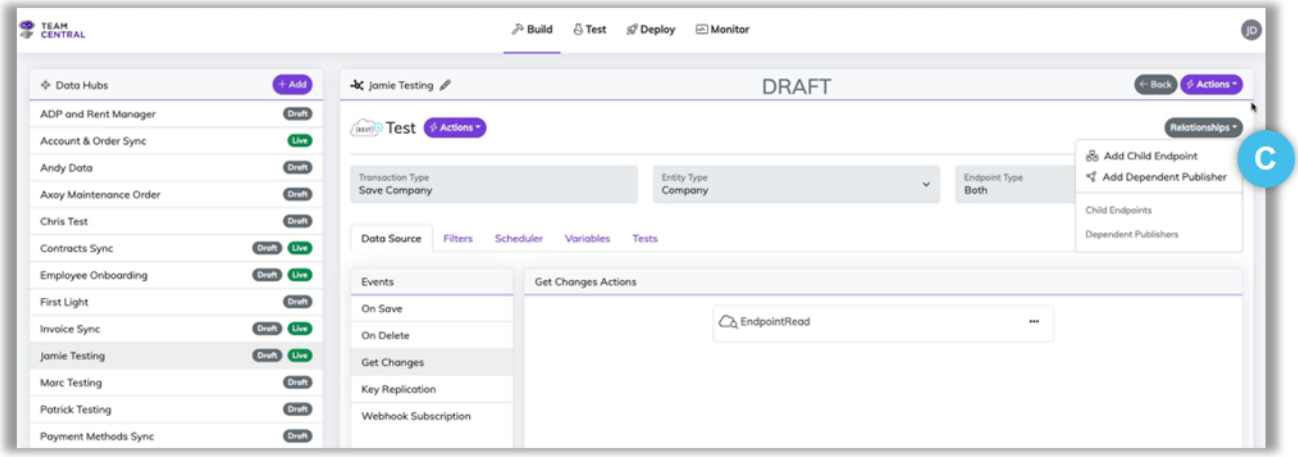
- Delete Publishers must be set-up as a Dependent Publisher for the Endpoint that publishing the data begins with. It is important to note that the Delete Publisher and the Delete Subscriber **must have the same Transaction Type** as their parent.
- When adding a Delete Publisher to any subscribing Endpoints, you must both set the Support Operations of the Endpoint to support deletes **and** you must configure the delete action to execute the delete.

Add a Child Endpoint

1. Select **Build** > **Data Hubs** from the main menu.
2. Select the appropriate **Data Hub** (A) in the left column
3. Select the appropriate parent **Endpoint** (B).



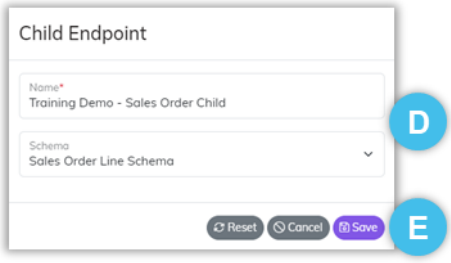
4. Select the **Relationships** drop-down menu; then, select **Add Child Endpoint** (C).



5. Enter a **Name** and select a **Schema** to link (D).

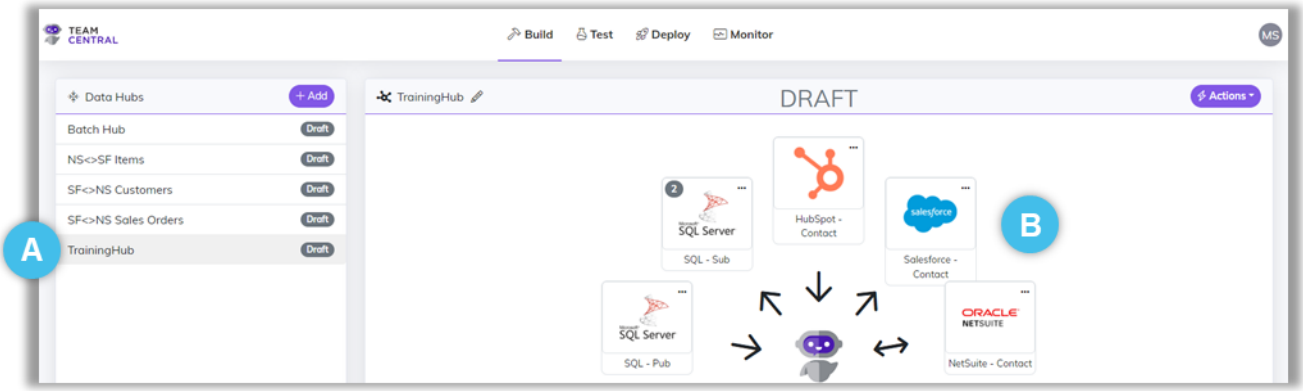
NOTE:	Only Schemas associated with the selected Endpoint will be selectable.

6. Select **Save** (E).

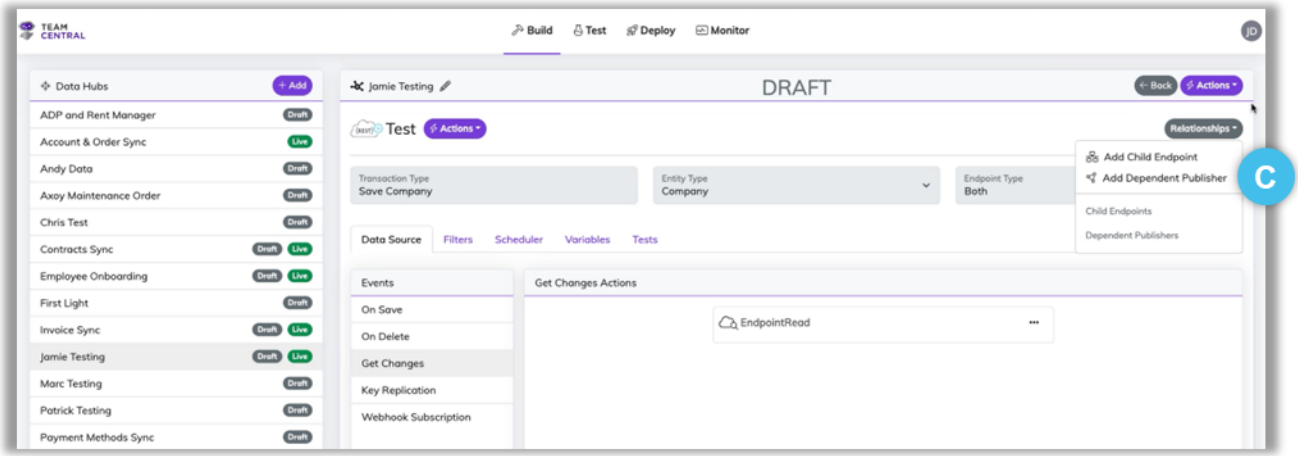


Add a Dependent Publisher

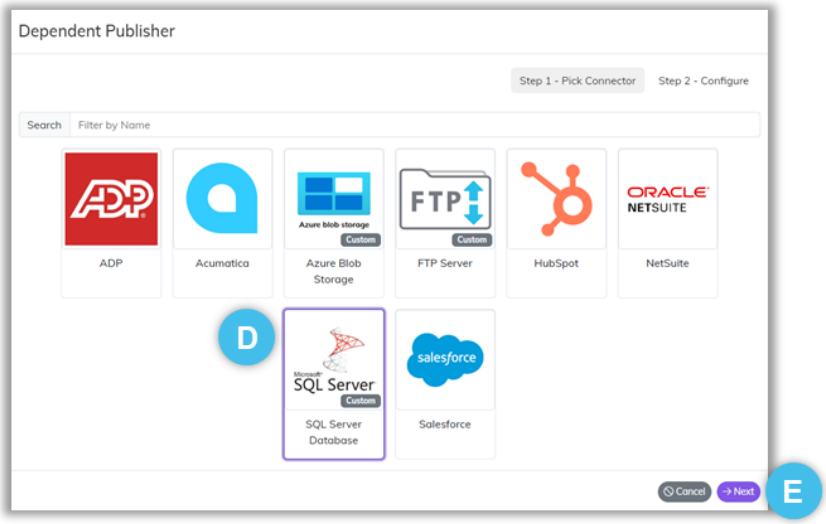
1. Select **Build** > **Data Hubs** from the main menu.
2. Select the appropriate **Data Hub** (A) in the left column
3. Select the appropriate parent **Endpoint** (B).



4. Select the **Relationships** drop-down menu; then, select **Add Dependent Publisher** (C).

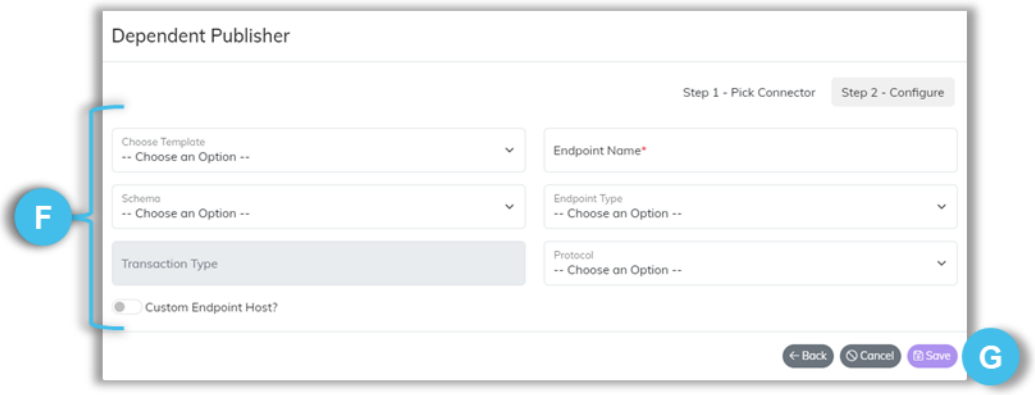


5. Select the Dependent Publisher **Connector** (D); then, select **Next** (E).



6. Configure the Dependent Publisher by entering the appropriate **data** (F); then, select **Save** (G).

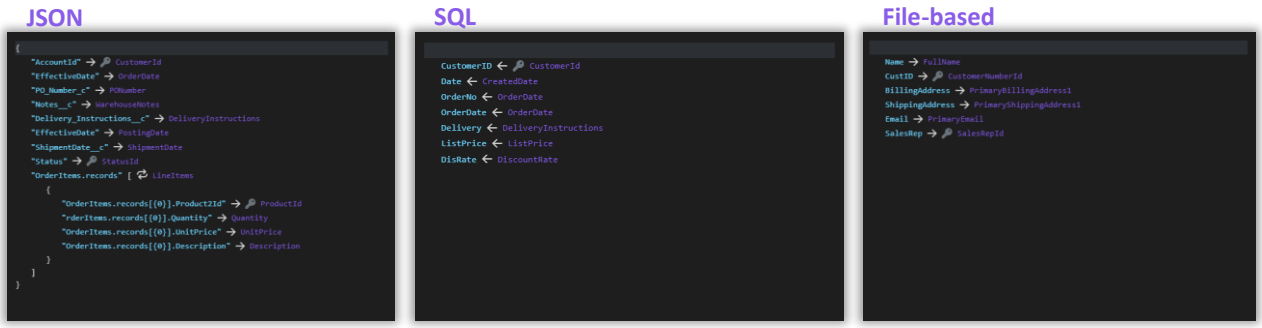
NOTE:	For additional instructions on configuring the publisher, see the *Add Endpoints* section of this user guide.



Map Endpoint Actions

A core tenet of Central is decoupled systems, so it is not necessary to map systems together in a point-to-point manner. Everything gets mapped to the Schemas that you created, both inbound and outbound of an Endpoint. This is done by mapping Endpoint Actions.

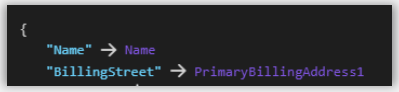
Mapping Endpoint Actions tells the Common Model exactly how to perform a particular task, such as *what data to get*, *where to get that data*, *where to publish that data*, *a method to read or write the data*, and a few other optional selections. This is where you tie all of your Schema Definitions to the Connector systems, Mapping the flow of data. Most Connectors utilize JSON or XML language to "talk" to the Common Model; however, the types of maps are dependent on the type of connector data you are mapping to (JSON, XML, SQL, and Delimited Files). As before, the Connector Property field must match the exact naming and language of the Connector.



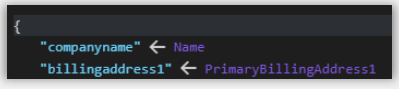
Example: HubSpot references website data as *domain*; whereas NetSuite references website data as *url*. The Common Model enables you to map both systems to the Schema Definition you created, *WebSite*. So, when the Common Model subscribes to data from HubSpot, it knows it's looking to read the *domain* field. Then, it transcribes that data through the *WebSite* field of the Common Model. When publishing that same data to NetSuite, it knows to reference the *url* field since you created that Mapping.

NOTE: When you open the Mapping screen, you will see a black code box on the left, with an open area to add properties on the right. The code box will display the details as you create the Mapping. The Connector data is always shown on the left side in blue text, while the Common Model data (Schema Definitions) is shown on the right side in purple text. The arrows between the Connector and Common Model indicate the direction data will flow. The arrow will always point towards the application receiving the data. Fixed values will always show in orange text.

For example, when a connection is publishing data to the Common Model via JSON protocols, the arrow will point from the Connector to the Common Model – as shown here.



However, when working on the subscribing side, the arrow will point towards the Connector – since the data is flowing from the Common Model to the Connector.

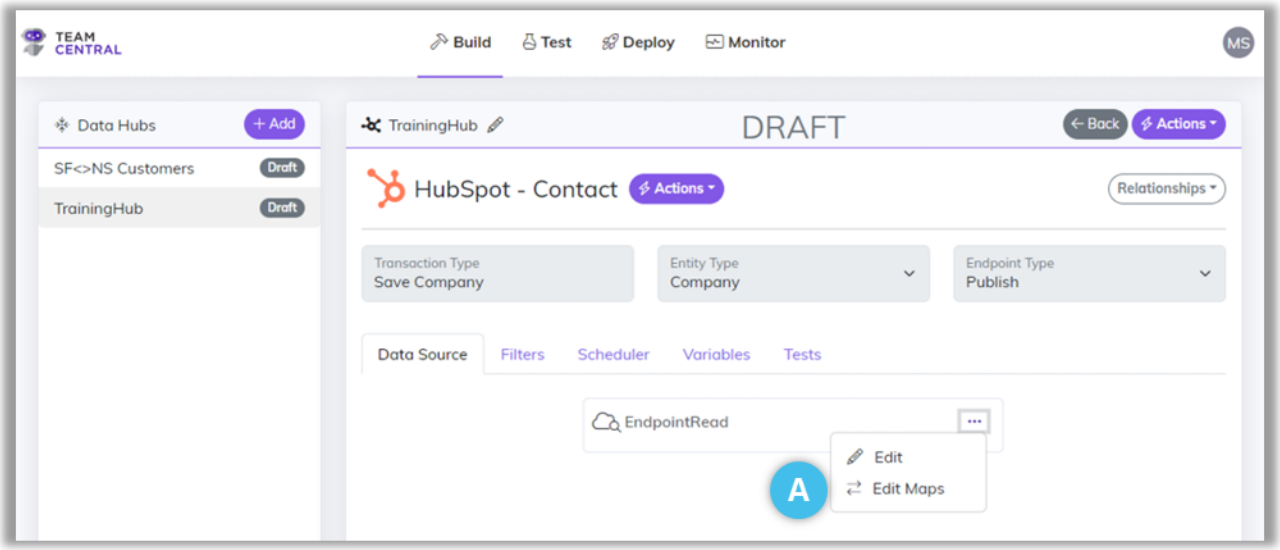


Advanced Tool: Advanced users can view, and in some cases modify, the underlying configuration that is fed into the engine. To access code view, select the code icon in the top right corner.

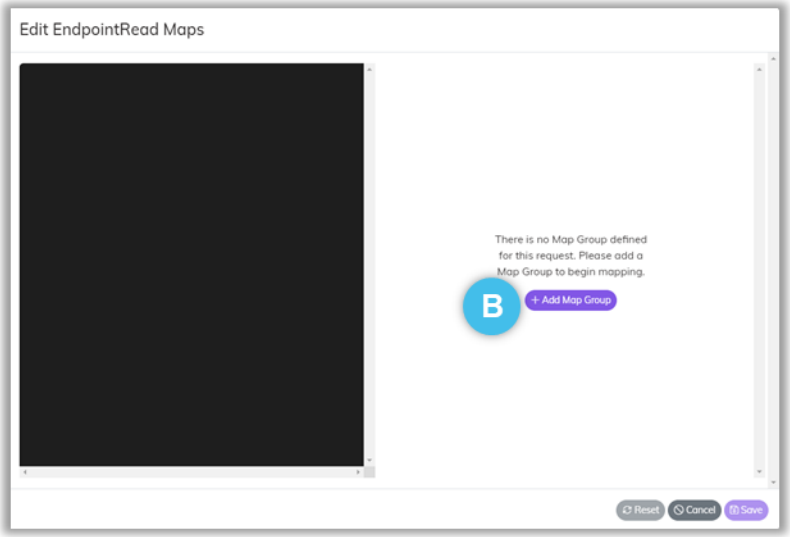
NOTES:

- The following instructions for Mapping Endpoint Actions are generically written for JSON protocols; however, each protocol will have its own distinct appearance and follow its own standards (e.g. JSON will present within curly brackets, while XML will present within square brackets).
- Additionally, these instructions guide you through the addition of a Simple Property. However, you will select the most applicable Property Type for your protocol and data; then, loosely follow the same instructions.
- **For detailed descriptions of mappable properties for each protocol**, see the ***Mapping Data on an Action*** table at the end of this section.

1. Select the appropriate **Data Hub** > **Endpoint**; and ensure you are on the **Data Source** tab.
2. Select the **menu** icon > **Edit Maps** (A).

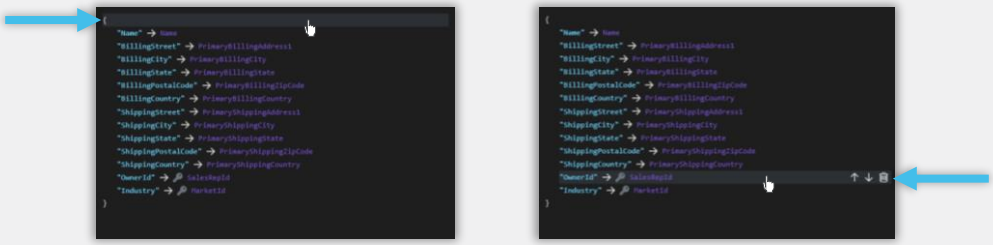


3. Select **Add Map Group** (B).



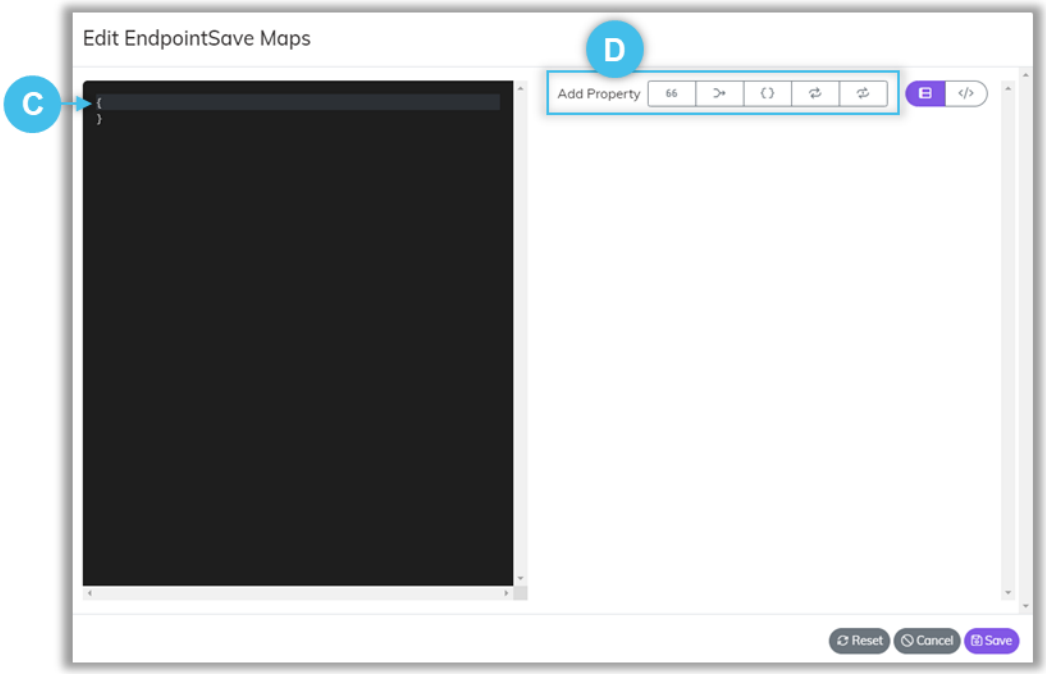
4. Select the **top row** (C) of the code box on the left side.

> **NOTE:** If you are simply adding a map to the route, then select the top line in the code box. However, when you want to add a map to a specific action (e.g. Saves), select that line in the left pane (specifically, where on the payload) you want to add your map.



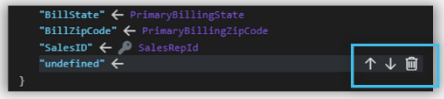
5. Select a means to map the data via the **Add Property** selectors (D).

> **NOTES:**
> - Only mappable options will be displayed based on your configured Endpoints.
> - You can add one or more of the following Property Types to a single Map.
> - For detailed descriptions of mappable properties for each protocol, see the **Mapping Data on an Action** table below.



6. Select a **Central Property** (E).
7. Define the **Connector Property** (F).

NOTES:	• To delete a map property, select the **delete** icon in that row of the code. • To adjust the order of a Mapping element, select the **up** or **down arrow** in that row of the code.



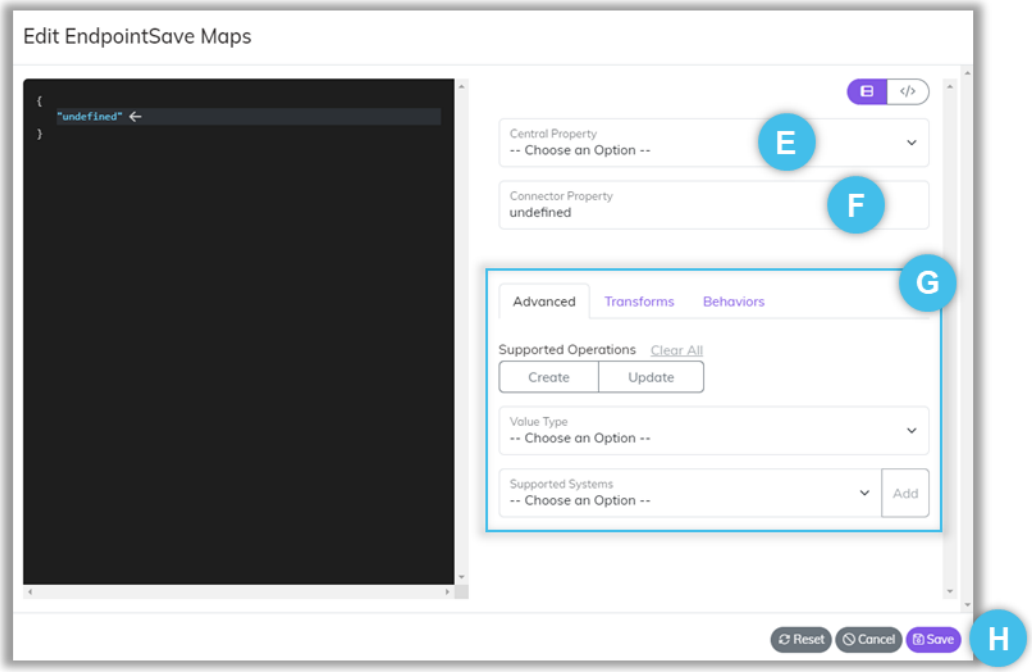
8. As needed, **adapt** the data (G).

NOTE:	It is possible to adapt the data as it is published to the Common Model or as an Endpoint subscribes to that data. For additional information and instructions, see the *Adapting Data when Mapping* section of this user guide.

9. Repeat the previous steps to construct all of the necessary Mapping.

⚠ **NOTE:**	While Mapping, select **Save** as needed; doing so will not close this window.

10. When finished, select **Save** (H). Then select **Cancel** to close the window.



Best Practice:	While it is possible to deploy from the Data Hubs screen, it is strongly advised not to perform this action here. It is recommended to deploy all data hubs from the **Deploy** > **Data Hubs** tool, where you can add comments, view the code, and compare this deployment with Live or Previously Deployed versions.

Mapping Properties of Endpoint Actions per Protocol

⚠ **NOTE:**	Users should be familiar with XPath to complete Read mapping.

Action	Mapping
JSON Reads	All JSON Read maps use JSONPath for the Connector Property to tell the mapping engine where on the payload we should map the data from. • **Property:** Maps a single value in a JSON document to a property on your Schema. • **Multi-Property:** Generally used in tandem with a Transform, it takes multiple values on a JSON document, transforms those values in some way, and then sets the transformed result onto a property on your Schema. • **Looping List:** Used to bind an array on the connector JSON payload to a list of items on your schema. For example, a Looping List can be used to map the lines of a Sales Order or the Prices of an Item.
JSON Saves	• **Property:** Maps a single value on your schema to property on the connectors Json input • **Multi-Property:** Generally used in tandem with a Transform, it can take multiple properties on your Schema and output a single property on the Connectors Json input. • **Nested Property:** Used to create a static Json nested property. Usually used in tandem with another map that is embedded inside of it. For example, in this input: { “customerName”: { “firstName”:”Marc”, ”lastName”:”Johnson”}}, the nested Property would be “customerName”. • **Static List:** Used to create a static Json array to insert static items, such as “itemPrices”:[{“price”: 23.12}]. • **Looping List:** Used to bind a dynamic list of items on your Schema list to an array of items on the connectors Json input.

XML Reads	Uses XPath to map the data from the connector to your Schema. It is important to note that the first step in creating an XML Read Mapping is to add the root Element of the XML Document. For example, in a SOAP document, the root is typically <Envelope></Envelope>. • **Simple Element:** Maps a single value in an XML Element or XML Attribute to a property on your Schema. As shown in the example below, to map the first name value, enter “FirstName.” To map from a nested value (e.g. street), enter "Address/Street.” In the example, XML title is an attribute to map the title value, enter "Person/@title." • **Multi-Element:** Generally used in tandem with a Transform, it can take multiple values on an XML document, transform those values in some way, and then sets the transformed result onto a property on your Schema (e.g. to combine a person’s first name and last name). • **Looping List:** Used to map a list of repeating items in the connector XML document to a list of items on your Schema. For example, a Looping List can be used to map the phone numbers (as shown below) into a list on the Schema. • **Syntax Example:** <Person title="Mr."> <FirstName>John</FirstName> <LastName>Doe</LastName> <Address> <Street>123 Main St</Street> </Address> <PhoneNumbers> <PhoneNumber type="home">555-1212</PhoneNumber> <PhoneNumber type="work">555-2323</PhoneNumber> </PhoneNumbers> </Person>
XML Saves	On an XML Save, your maps will be used to build the XML document that gets sent to the Connector. Every layer of the XML document needs to be built and accounted for in your mapping. You will be mapping both structure and content. • **Element:** Adds an element to the XML with the value selected from the Schema. • **Attribute:** Attributes can be added to any Element and will map to the value selected in the Schema. • **Multi-Element:** Generally used in tandem with a Transform, it can take multiple properties on your Schema and output a single Element on the connectors Xml input. • **Nested Element:** Used to create a static XML nested Element. Typically used in tandem with another map that is embedded inside of it. For example, in this input: <customerName><firstName>Marc</firstName><lastName>Johnson</lastName></customerName>, the nested Element would be <customerName>. • **Static List:** Used to create a static list of XML items where you can insert static items. For example: <itemPrices><price>23.12</price></itemPrices>. • **Looping List:** Used to bind a dynamic list of items on a list on your Schema, to a list of items on the Connectors XML input.

SQL Reads	• **Column:** Maps a single value from a database to a property on your Schema. • **Multi-Property:** Generally used in tandem with a Transform, it can take multiple database columns, transform those values in some way, and then sets the transformed result onto a property on your Schema.
SQL Saves	• **Property:** Maps a single value on your Schema to a column on a database. • **Multi-Property:** Generally used in tandem with a Transform, it can take multiple properties on your Schema and output a single column on the database.
File Reads	• **Column:** Maps a single value from a file to a property on your schema. • **Multi-Property:** Generally used in tandem with a Transform, it can take multiple file columns, transform those values in some way, and then set the transformed result onto a property on your Schema.
File Saves	• **Property:** Maps a single value on your Schema to a column on a character delimited file. • **Multi-Property:** Generally used in tandem with a Transform, it can take multiple properties on your Schema and output a single column on a file.

Adapting Data when Mapping

There are ways to adapt the data being read or written by the Common Model. For example, if one system displays data as a decimal number, but you want the data to show as a monetary value, you can transform that data as the Common Model reads and/or writes it. Adapting data is more likely to be done on the subscribing side, but is possible to execute on both.

⚠ **NOTE:**	Only available adaptation options will be displayed based on your configured Endpoints and Mapping Actions.

Advanced Tab

In the advanced settings, you can dictate when data gets transferred from the Common Model to other systems. For instance, on the subscribing side, you may want some functionality to only execute on when a new record is created or only when data is updated. Also, data taken from the Common Model that is a string, but needs to be written to a Connector as a double or decimal value, changing the output, can be done here.

Action	Mapping
Read Action	• **FK System Override:** When the data that is read from the Connector creates a Foreign Key on the Schema, it will by default set the system of Connector. This override allows you to override the key as a value from another system.
Write Action	• **FK System Override:** When the data is written to the Connector, it will by default pull the Foreign Key value of the Connector it is writing to. This override allows you to override the key as a value from another system. • **Supported Operations:** By default, the map will execute for both Create and Updates; however, Supported Operations allows you to direct the map to only execute on a specific Write Operation, Create or Update. • **Value Type:** Allows you to format the data for a given type numeric, date, etc. • **Supported Systems:** By default, the map will execute for any system that posts a message that this Connector receives; however, Supported Systems allow you to only execute this map if the message came from a specific list of systems. (**Note:** Users should implement this functionality infrequently and only in necessary instances, as this will isolate the data transfer to that specific Mapping.) • **FK Use Label:** By default, the Foreign Key map writes the value of the Foreign Key; however, this setting allows you to override the default and instead use the Label of the Foreign Key that is stored inside the Central Index.

Transforms Tab

Transforms can be added to either subscribe or publish Mapping; however, are more commonly used to modify the data being consumed by a system (on the subscribe side). Transforms can be strung together to manipulate data in multiple ways.

Action	Description
Addition	• Adds a value to a property on the Schema and outputs the result. • **Inputs:** Add – (the value to be added)

BizLogicMap	• Adds conditional logic to a single property.
	• Within a BizLogicMap, you can add Groups for more complex scenarios, where you can achieve And/Or scenarios, but between groups it is supports only Or. For example, if the property value equals X or the property value equals Y, then return Z.
	• **Inputs:**
	  ○ Group: And/Or, Operator, Value, Return Value
	  ○ Else Return Value: If all of the groups return false, this value is returned from the map

BizLogicMap – Simple	• Adds conditional logic to a single property. This is the same concept as BizLogicMap, but only matches for equality and does not allow for more complex logic with groupings (e.g. if equals this value then return that value).
	• **Inputs:**
	  ○ Comparison: The value you are comparing the property to be equivalent to; then Return: If the value of the property equals the value of the Comparison, then return this value
	  ○ Return Null, Return Original Value, Else Return Value

Compute Expression	• This transform can be used to take in multiple properties and perform conditional logic or computation and return a value. • **Link to documentation:** Expression Property – .NET | Microsoft Learn. • **Inputs:** Expression (uses the syntax for .NET DataTable.Compute)
Convert UTCtoTimeZone	• Used to convert a date time that is in UTC format to a specific time zone, most likely on a subscriber. • **Inputs:** Time Zone
CsvToList	• A single property that is a character delimited list of values, split into a List. • This transform will only work when mapping into values on the Common Model that is a list of strings. • **Inputs:** Make Distinct, Trim Items, Character Separator
DefaultToCurrentDate	• If the incoming property value coming is null, it will default to UTC date/time. • Typically used on saves for a create record. • **Inputs:** None

DefaultValue	• If the incoming property value coming is null, it will default to this value**.** • Typically used on saves for a create record. • **Inputs:** A predetermined value
Division	• This takes a property on the Schema and divides by a value; then, outputs the result. • **Inputs:** A Divisor (the value of what you are dividing by)
DateTimeFlattening	• Used to break a date apart into individual properties on a Save, or to combine them on a Read. • JSON Example: `{ "year": 2023, "month": 1, "day": 1 }` • **Inputs:** Include Time, Protocol, Year, Month, Day
FormattingTemplate	• Takes one to many properties on the Schema and does a string format; then, returns the result**.** • **Inputs:** Template ○ The format of the *what* you want returned. If you have three properties that are being passed into the transform, and you want a hyphen between each property, the Template would be {0}-{1}-{2}. The mapping engine does a replace on the {0} with the value of the first property being passed into the Template, and so on with the second and third properties.
JSONContains	• Executes a JSONPath. If it finds a value, it returns *true*, else returns *false* • **Inputs:** JSONPath
MathTruncate	• Trims off all decimal values. • **Inputs:** None
Multiplication	• Multiplies a value to a property on the Schema and outputs the result. • **Inputs:** Multiplier (the value to be multiplied)
RegexMatch	• Pulls a value out of string based on a pattern; then, Returns text from within a string that matches the provided regular expression. • Only the matching text will be used in the map (to find specific text with a string, such as a specifically formatted date that you want to pull just the month). • **Inputs:** Match Pattern
RegexReplace	• Formats a string of digits into a specific format; for example, formatting 10 digits into a specific phone number format. • **Inputs:** Match Pattern, Replace Pattern (The Match Pattern tells it what to find in the string, and the Replacement Pattern tells it how to reformat the string based on what has matched.)

Round	• Applies rounding on a numeric property.
	• **Inputs:**
	  ○ Behavior: Always round up, Always round down, Default
	  ○ Precision

Substring	• Retrieves a portion of a string. For example, if a field can only accept 30 characters, Substring can be used to return only the first 30 characters of a property and remove the rest.
	• **Inputs:**
	  ○ Starting Index: the position on the string you want to start with (e.g. the starting position is 0)
	  ○ Number of Characters

Subtraction	• Subtracts a value from a property on the Schema and outputs the result • **Inputs:** Subtract – (the value being subtracted)

Behaviors Tab

Behaviors are advanced level options to assist developers in further isolating data transfers. They are unique overrides, used to adapt the data so that it behaves differently than it typically would.

Action	Description
Allow Empty Fixed Value	• Used primarily on Write Maps, it sets empty string on a field that we do not have mapped, but that needs to be included on the payload of a Save.
Allow Primary Key to Change	• Used primarily on Read Maps to tell the engine it can update a value on a Primary Key in the Central Index. • By default, Central prevents this from happening, so this acts as an override.
Map from Absolute Root	• Used primarily on Read Maps, and mainly inside of nested maps, to path to the top level.
Skip if Null	• Used primarily on Write maps, in instances where you want to ignore the map entirely if the incoming value on the property is null.

Additional Data Hub Adaptation Methods

Publish/Subscribe Filters

A Publish or Subscribe Filter enables you to create rules to build complex scenarios to determine *what* data gets published. You can create individual filters, as well as groups of filters; and then configure And / Or conditions at both the Group- and Item-level, as well as across groups and items.

For Publish Endpoints, some connectors like Salesforce, Microsoft, and Netsuite have powerful query capabilities inside of the API to perform complex filter logic at the API level; other Connectors do not have this capability. A Publishing Filter can be used to filter the payload that has been downloaded by a Connector, manipulating *what* data is published.

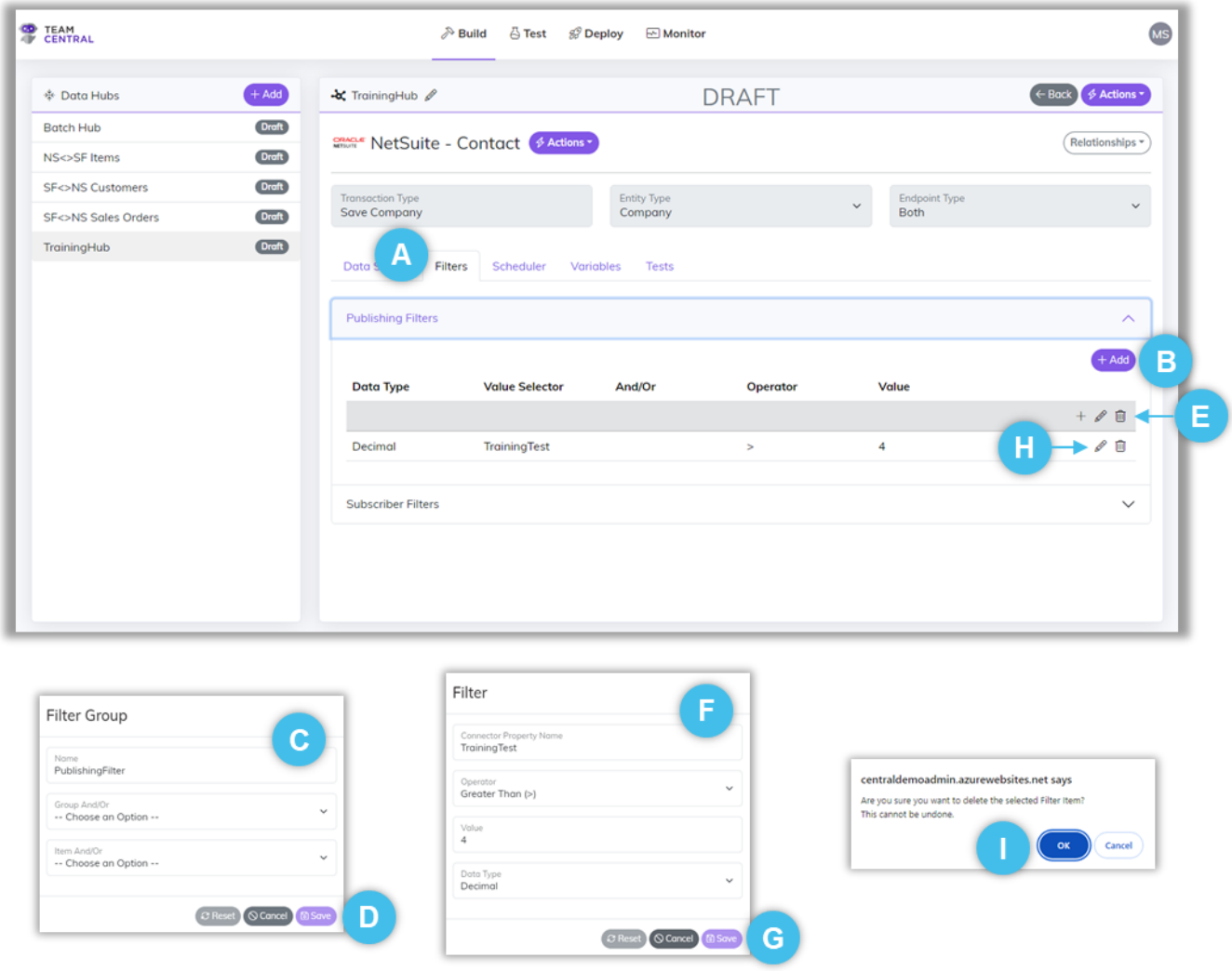
For Subscriber Endpoints, it allows an Endpoint to receive a message, but not process it based on the data that is received on the message.

- **Publishing Filter:** Uses JSONPath or XPath to select a field from the Connector to use a comparison in the filter. (This is the same syntax as on an XML Simple Element Read mapping.)
- **Subscriber Filter:** Uses JSONPath to select a field from your Schema to use as a comparison in the filter. For example, to filter on a customer data value, use a JSONPath such as *CustomData[?(@.Name == 'Subscription3TimesRenewed')].Value*.

⚠ **NOTE:**	For additional help executing this functionality, contact TeamCentral Support.

1. Select the appropriate **Data Hub** > **Endpoint**; and ensure you are on the **Scheduler** tab (A).
2. Then, use the below table to determine next steps:

To:	Do this:
Create a New Filter Group	• Select **+ Add** (B). • Enter a **Name**; then, select a **Group And/Or**, an **Item And/Or**, and a **Subscriber Process Stage** (*subscriber filters only*) (C). • Then, select **Save** (D).
Add a New Item within a Filter Group	• Select the **+ icon** (E) in the top row of the Filter Group. • Enter or select the appropriate data for **Connector Property Name**, **Operator**, **Value**, and **Data Type** (F). • Then, select **Save** (G).
Edit a Filter Group	• Select the **edit** icon (E) in the top row of the filter table. • Edit the necessary fields and select **Save**.
Edit an Item	• Select the **edit** icon (H) in the appropriate item row of the filter table. • Edit the necessary fields and select **Save**.
Delete a Filter Group	• Select the **delete** icon (E) in the top row of the filter table. • Select **OK** to confirm (I).
Delete an Item	• Select the **delete** icon (E) in the appropriate item row of the filter table. • Select **OK** to confirm (I).


TEAM CENTRAL
Build
Test
Deploy
Monitor
MS
Data Hubs
+ Add
Batch Hub
Draft
NS<>SF Items
Draft
SF<>NS Customers
Draft
SF<>NS Sales Orders
Draft
TrainingHub
Draft
TrainingHub
DRAFT
Back
Actions
NetSuite - Contact
Actions
Relationships
Transaction Type
Save Company
Entity Type
Company
Endpoint Type
Both
A
Filters
Scheduler
Variables
Tests
Publishing Filters
+ Add
B
Data Type
Value Selector
And/Or
Operator
Value
E
Decimal
TrainingTest
>
4
H
Subscriber Filters
Filter Group
C
Name
PublishingFilter
Group And/Or
-- Choose an Option --
Item And/Or
-- Choose an Option --
Reset
Cancel
Save
D
Filter
F
Connector Property Name
TrainingTest
Operator
Greater Than (>)
Value
4
Data Type
Decimal
Reset
Cancel
Save
G
centraldemoadmin.azurewebsites.net says
Are you sure you want to delete the selected Filter Item?
This cannot be undone.
I
OK
Cancel

Scheduler

A Scheduler is available as a Publisher option only, and can be applied to schedule *when* data gets published.

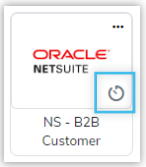
There are several ways to schedule the data to be published:

- **On Demand:** Runs manually when the user decides to run.
- **Interval:** Runs on a specific frequency based on Minutes and Seconds.
- **Day of the week:** Runs on specific days of the week based on Time Zone, Days of the Week selected, Time (Hours and Minutes).
- **Day of the month:** Runs on specific days of the month based on Time Zone, Days of the Month, Time (Hours and Minutes), Run Conditions (Any Day, Week Day Only Advance – if day lands on weekend advance to next business day, Week Day Only Skip – if day lands on a weekend then skip entirely, Week Day Only)

Best Practice:

When a Scheduler has been applied to an Endpoint, a **scheduler** icon will be shown on the Endpoint within the Data Hub.

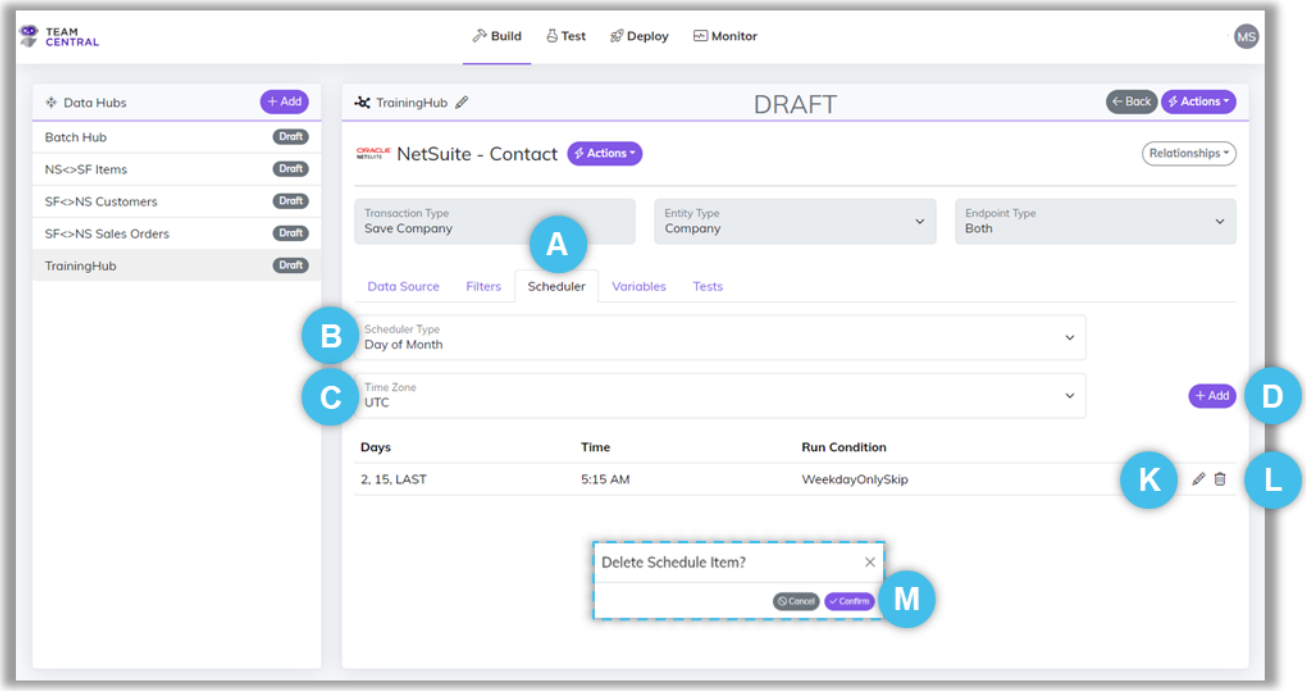
Users can select to manually publish the Endpoint as needed, but it is advised to allow the scheduler to run as prescribed. This will help to avoid any duplicates and/or errors.

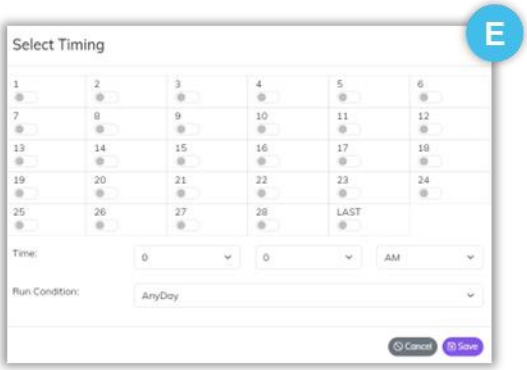


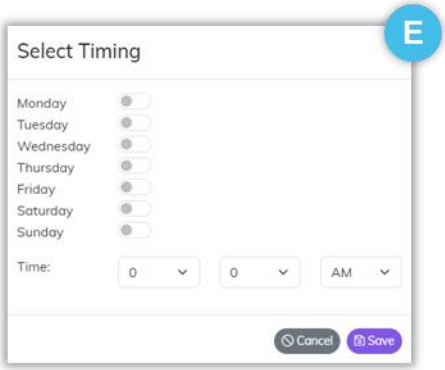
---

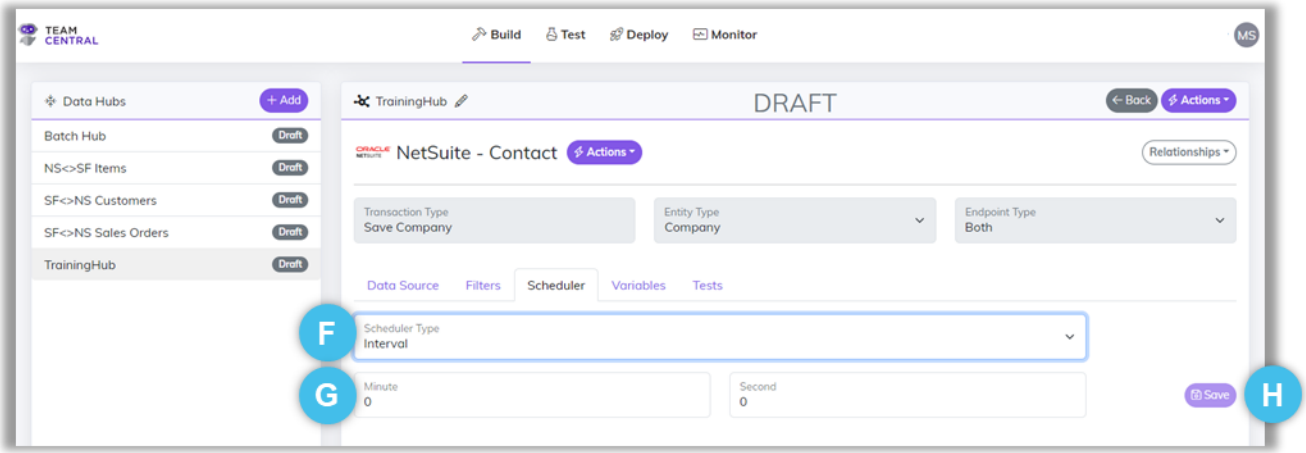
1. Select the appropriate **Data Hub** > **Endpoint**; and ensure you are on the **Scheduler** tab (A).
2. Then, use the below table to determine next steps:

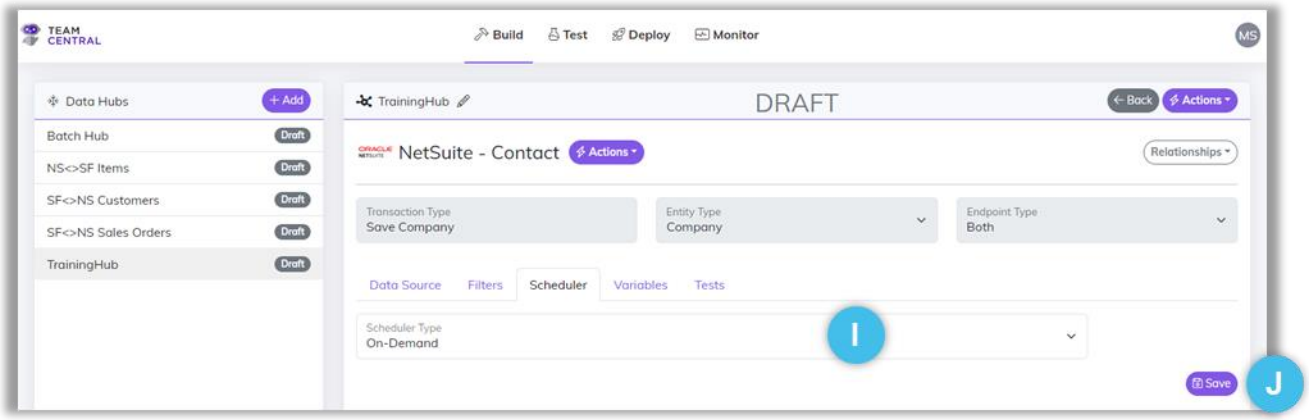
To:	Do this:
Create a New Day of the Week or Day of the Month Schedule	• Select the **Scheduler Type** (B). • Select a **Time Zone** (C). • Select **+ Add** (D). • Select the appropriate **day**, **time**, and **Run Condition** (*monthly schedule only*); then, select **Save** (E).
Create an Interval Schedule	• Select the **Scheduler Type: Interval** (F). • Enter the **Minute** and **Second** times (G). • Then, select **Save** (H). **Note:** To edit an Interval Schedule, simply change the **Minute** and **Second** intervals; then select **Save**.
Create an On-Demand Schedule	• Select the **Scheduler Type: On-Demand** (I). • Then, select **Save** (J).
Edit a Day of the Week or Day of the Month Schedule	• Select the **edit** icon (K) in the appropriate row. • Edit the **Schedule** as needed; then, select **Save**.
Delete a Day of the Week or Day of the Month Schedule	• Select the **delete** icon (L) in the appropriate row. • Select **Confirm** (M).


TEAM CENTRAL
Build
Test
Deploy
Monitor
MS
Data Hubs
Add
Batch Hub
Draft
NS<>SF Items
SF<>NS Customers
SF<>NS Sales Orders
TrainingHub
TrainingHub
DRAFT
Back
Actions
NetSuite - Contact
Relationships
Transaction Type
Save Company
Entity Type
Company
Endpoint Type
Both
A
Data Source
Filters
Scheduler
Variables
Tests
B
Scheduler Type
Day of Month
C
Time Zone
UTC
Add
D
Days
Time
Run Condition
2, 15, LAST
5:15 AM
WeekdayOnlySkip
K
L
Delete Schedule Item?
Cancel
Confirm
M


Select Timing
E
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
LAST
Time:
0
0
AM
Run Condition:
AnyDay
Cancel
Save


Select Timing
E
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Time:
0
0
AM
Cancel
Save


TEAM CENTRAL
Build
Test
Deploy
Monitor
MS
Data Hubs
Add
Batch Hub
Draft
NS<>SF Items
SF<>NS Customers
SF<>NS Sales Orders
TrainingHub
TrainingHub
DRAFT
Back
Actions
NetSuite - Contact
Relationships
Transaction Type
Save Company
Entity Type
Company
Endpoint Type
Both
Data Source
Filters
Scheduler
Variables
Tests
F
Scheduler Type
Interval
G
Minute
0
Second
0
Save
H


TEAM CENTRAL
Build
Test
Deploy
Monitor
MS
Data Hubs
+ Add
Batch Hub
Draft
NS<>SF Items
Draft
SF<>NS Customers
Draft
SF<>NS Sales Orders
Draft
TrainingHub
Draft
TrainingHub
DRAFT
Back
Actions
NetSuite - Contact
Actions
Relationships
Transaction Type
Save Company
Entity Type
Company
Endpoint Type
Both
Data Source
Filters
Scheduler
Variables
Tests
Scheduler Type
On-Demand
I
Save
J

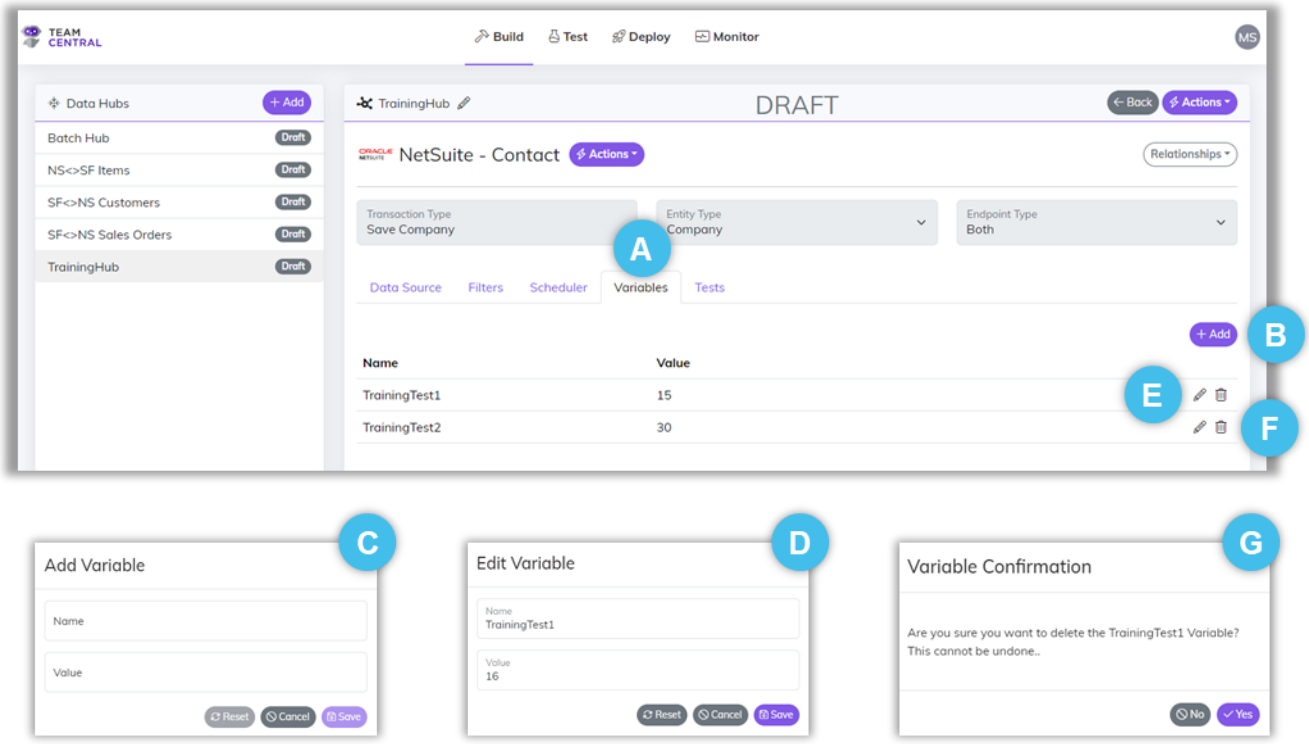
Variables

Variables enable you to create values that can be passed into the action and used as parameters for a URL or message body. For custom Endpoints, variables can be used as constants that can be referenced in code as an application setting.

NOTE:	For additional help executing this functionality, contact TeamCentral Support.

1. Select the appropriate **Data Hub** > **Endpoint**; and ensure you are on the **Variables** tab (A).
2. Then, use the below table to determine next steps:

To:	Do this:
Create a New Variable	• Select **+ Add** (B). • Enter a **Name** and/or **Value**; then, select **Save** (C).
Edit a Variable	• Select the **edit** icon (D) in the appropriate row. • Edit the **Name** and/or **Value**; then, select **Save** (E).
Delete a Variable	• Select the **delete** icon (F) in the appropriate row. • Select **Yes** (G) to confirm.



Drop-Down Lookups

Drop-down Lookups are a specific type of cross-reference within Central, and are confined to a single Data Hub. They are typically associated with Foreign Keys that do not fit into a particular entity type on the Common Model, so they are classified as General System Lookups.

When a key is received by the mapping engine for a particular value, it goes to the list of Drop-Down values that match the value for that system. Then, it uses the name of the Drop-Down value to pull in all other Drop-Down values that match that name.

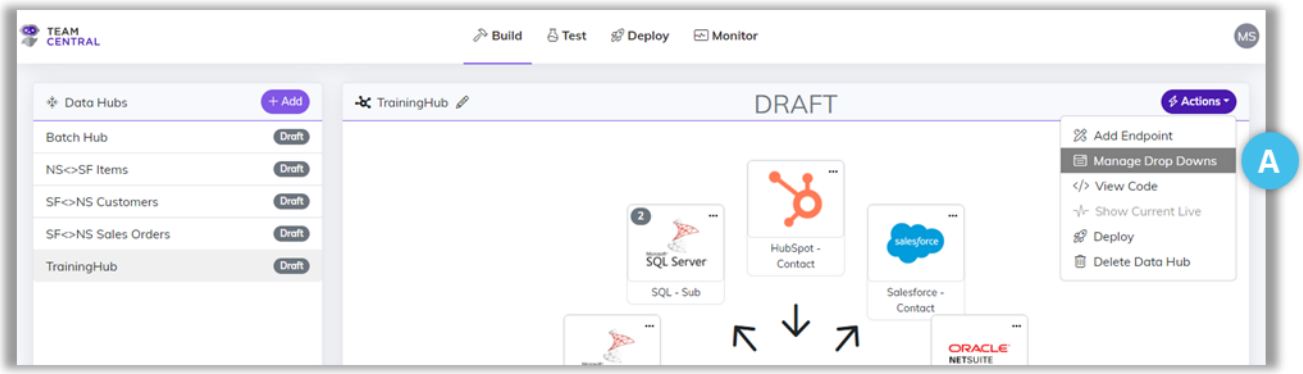
> **Example:** You may have an office location in Northwest Illinois. In SalesForce, that data might be written as NW-IL; however, in HubSpot, it may be written as a random string, 779f26-ghn11-ec145. You can build a Drop-Down Lookup to manage the cross-reference of this data.

A few important callouts on Drop-Down Lookups:

- The name of the Drop-Down field must match the name of the Foreign Key on your Schema in order for it to work.
- Drop-Downs are scoped (limited) to a specific Data Hub.
- An item inside of a Drop-Down list includes the name of the system, the name of the drop-down, and the value of the drop-down.
- The mapping engine uses the name and the system values of each Drop-Down item for the cross reference.

---

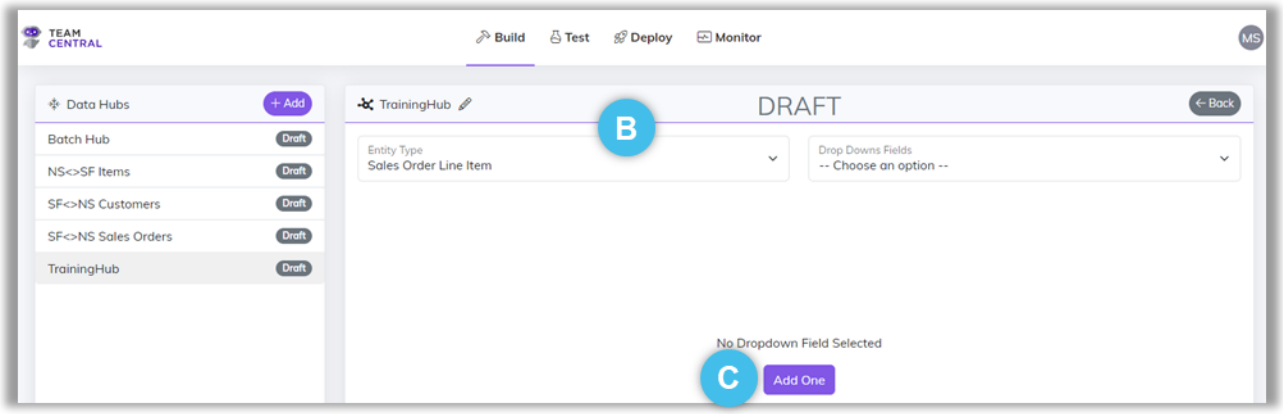
1. Select the appropriate **Data Hub** > **Endpoint**.
2. Select **Actions** > **Manage Drop Downs** (A).

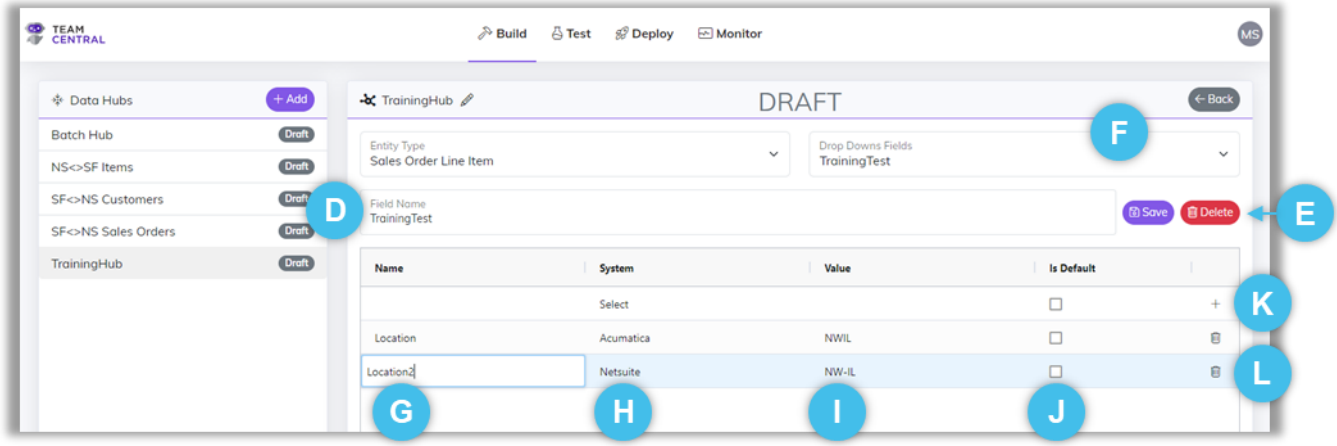


3. Then, use the below table to determine next steps:

To:	Do this:
Create a New Drop-Down	• Select an **Entity Type** (B). • Select **Add One** (C). • Enter a **Field Name** (D). • Select **Save** (E).

Add a new value	• Select the appropriate **Entity Type** (B) and **Drop-Down Fields** (F). • Select the **Name** cell (G) and enter the appropriate text. • Select the **System** cell (H) and choose the appropriate System from the drop-down list. • Select the **Value** cell (I) and enter the appropriate text. • As needed, check the **Is Default** box (J). • When finished, select **Save** (E).
Add additional new value(s)	• Select the appropriate **Entity Type** (B) and **Drop-Down Fields** (F). • Select the **+** icon (K). • Complete the steps to add a new value; and then, select **Save** (E).
Edit a value	• Select the appropriate **Entity Type** (B) and **Drop-Down Fields** (F). • Edit the data in the appropriate **field**; then, select **Save** (E).
Delete a value	• Select the appropriate **Entity Type** (B) and **Drop-Down Fields** (F). • Select the **delete** icon (L) in the appropriate row. • Then, select **Save** (E).
Delete a Drop-Down	• Select the appropriate **Entity Type** (B) and **Drop-Down Fields** (F). • Select **Delete** (E).





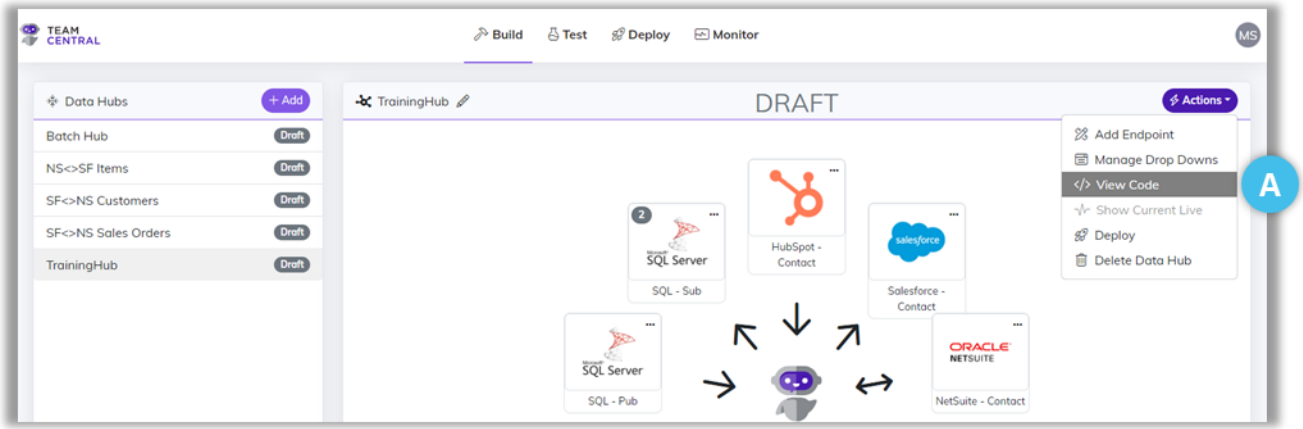
View Code

At the top level of the Data Hub, and for each individual map, you can flip to **Code View**, which shows the configuration that will feed into the mapping engine.

Advanced Configuration:	Code View can be used for advanced features that the UI does not currently support, but that can be configured by the TeamCentral support team.

Advanced Tool:	A Data Hub in Draft status can be modified and saved within the View Code feature.

1. Select the appropriate **Data Hub** > **Endpoint**.
2. Select **Actions** > **View Code** (A).



3. Edit the **code** (B) as needed.
4. Select **Save** (C).

Versioning

There are three (3) configuration versions that a Data Hub or Data Provider could be saved in:

Status	Definition
Draft	• The Data Hub/Provider is a work-in-progress – currently in the design, build, and/or testing phase. **Note:** Both a Live and Draft status can be held simultaneously. When modifications are made to a Live Data Hub, a draft copy is created to design, build, and test the connections prior to publishing. Once published, the draft/modified version will supersede the current live version. You can view the version history by selecting **Deploy** > **Data Hubs**; then select the applicable Data Hub in the left column.
Live	• The Data Hub/Provider is live in the Production environment today. • Live versions are read-only; a draft version must be created to modify (see note in Draft status definition).
Previous Deploy	• The Data Hub/Provider was in Production environment, but had to be taken down for additional design, building, and/or testing.

Deployment

Deploy a Data Hub

Best Practice: It is highly recommended to include comments every time you release a new version, to document any changes made and provide context between users.

1. Select **Deploy** from the main menu.
2. Select **Data Hubs** (A) from the submenu.
3. Select the appropriate **Data Hub** (B) from the left column.
4. Then use the below table to determine next steps.

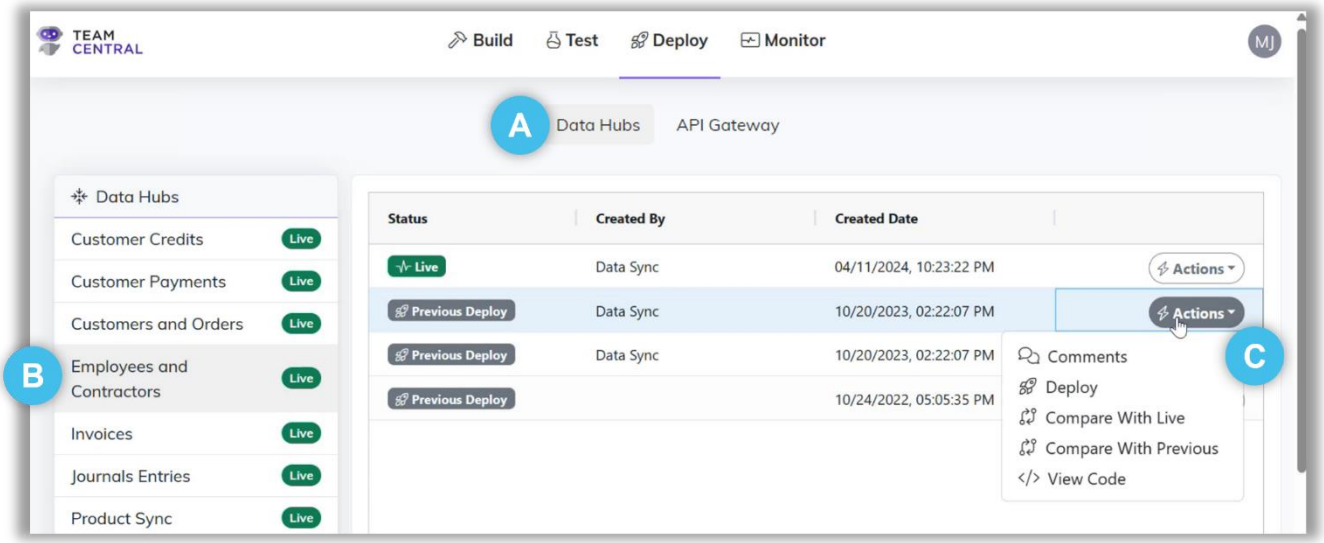
NOTE: You may perform several of the following steps before continuing to deploy. These actions can be taken on versions in any status; however, only available actions will be selectable.

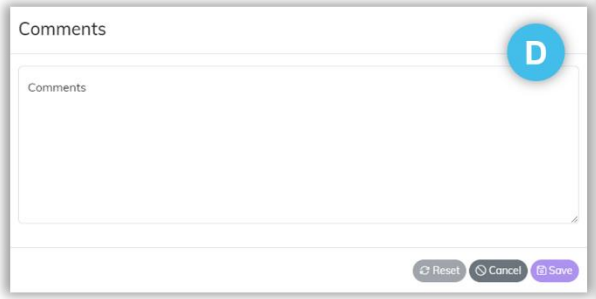
To:	Do this:
Add Comments	• Select **Actions** > **Comments** (C). • Enter text in the **Comments** field (D); then, select **Save**.
Compare with Live	• Select **Actions** > **Compare with Live** (C). • In the **Compare Configurations** window (E), view the side-by-side code for any changes and to ensure accuracy. • When finished, select **Cancel**.
Compare with Previous	• Select **Actions** > **Compare with Previous** (C). • In **Compare Configurations** window (E) view the side-by-side code for any changes and to ensure accuracy. • When finished, select **Cancel**.
View Code	• Select **Actions** > **View Code** (C). • View the code to ensure accuracy; when finished, select **Cancel** (F).

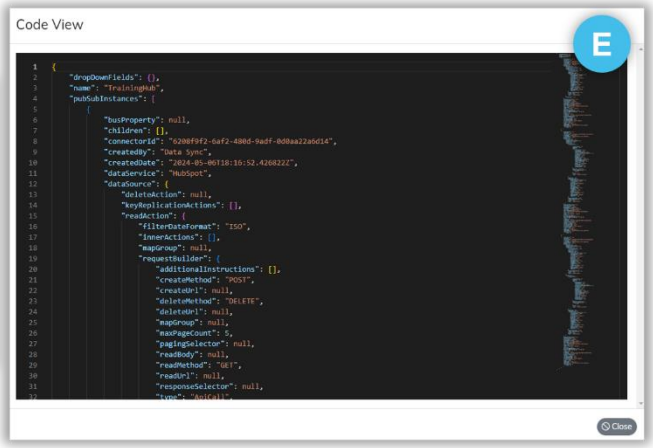
5. When ready, select **Actions** > **Deploy** (C) in the appropriate Data Hub row.
6. Select **Yes** (G) to confirm.

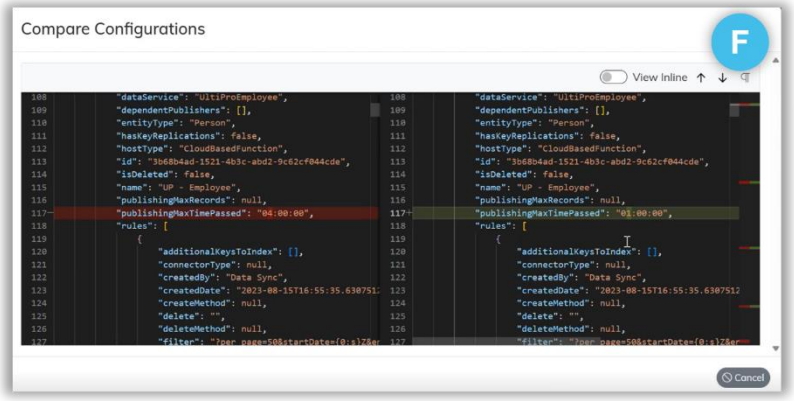
NOTE: Versions in either Draft or Previously Deployed status can be deployed. Both will supersede the current Live version.

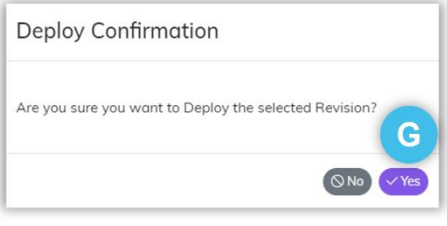
Example: If a newly deployed Data Hub inadvertently caused issues or didn't run properly, you can restore a Previously Deployed version to ensure the integration continues to run while you work on repairing the new release.


TEAM CENTRAL
Build
Test
Deploy
Monitor
MJ
A
Data Hubs
API Gateway
Data Hubs
Customer Credits
Customer Payments
Customers and Orders
Employees and Contractors
Invoices
Journals Entries
Product Sync
Live
B
Status
Created By
Created Date
Data Sync
04/11/2024, 10:23:22 PM
Actions
Previous Deploy
10/20/2023, 02:22:07 PM
10/24/2022, 05:05:35 PM
C
Comments
Deploy
Compare With Live
Compare With Previous
View Code


Comments
D
Comments
Reset
Cancel
Save


Code View
E
Close


Compare Configurations
F
View Inline
Cancel


Deploy Confirmation
Are you sure you want to Deploy the selected Revision?
G
No
Yes

Monitor a Data Hub

You can evaluate the health of your systems via the Monitor tool. The Dashboard displays the data in a macro view, showing the total number of successful data transfers versus errors occurring within each deployed Data Hub. You can view this information in either graph format or in a table format (which shows a more granular level of behavior). The Dashboard is view-only.

If you want to further investigate an error shown on the Dashboard, you can view that information in the Error Logs or Message Search. The Error Logs feature enables users to view a list of all unsuccessful actions. It pulls Receipts data (found within code view) into a quick and convenient table, displaying the most important information, including the Endpoint, an Identifier, Error Text, and Message Date.

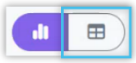
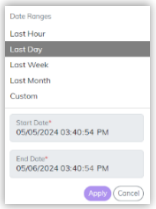
The Message Search tool provides both high-level and individual details for each transaction. You can both re-run the message (e.g. if you know the error was fixed and the system will publish accurately this time), and view the details of that error message to help you better assess the issue.

NOTE:	When an action fails, it typically identifies the error via a Main Key, which is sometimes an unidentifiable string of letters and/or numbers (i.e. GUID). Central has added an **Identifier** to help decipher this information. At times, the **Identifier** and Main Key may be the same data, such as in the case of a sales order number.

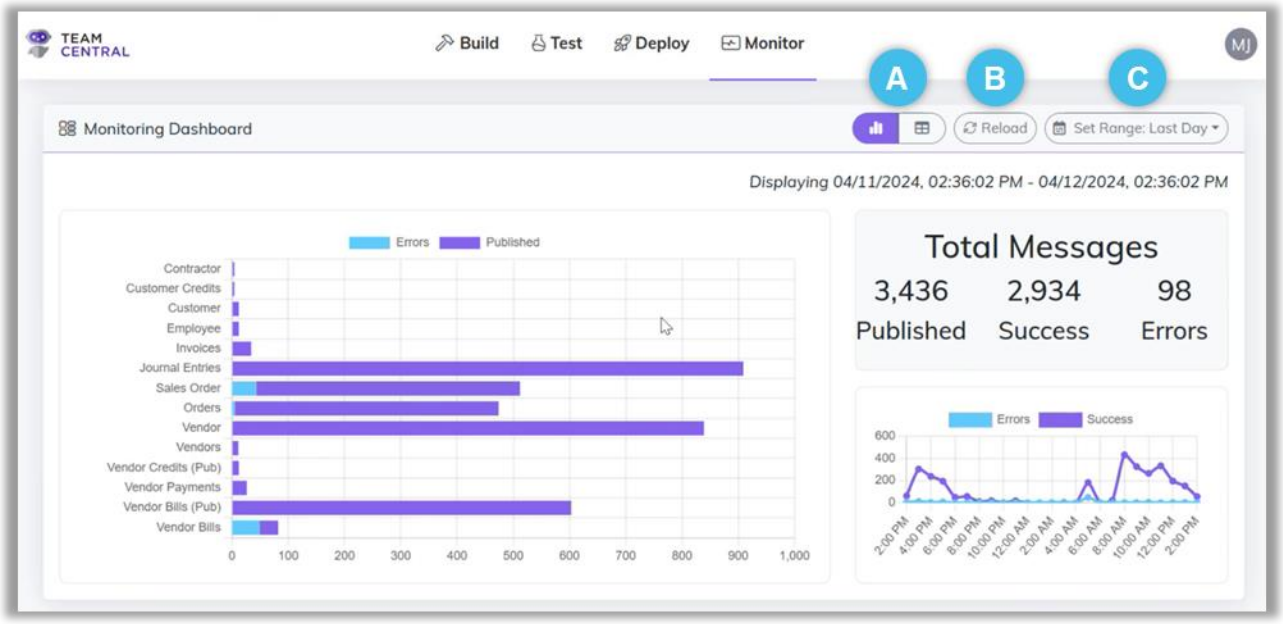
NOTE:	All information within the Monitor tool is view-only, with the exception of the Re-Run functionality within Message Search.

View the Dashboard

1. Select **Monitor** > **Dashboard** from the main menu.
2. Then, use the below table to determine the next steps.

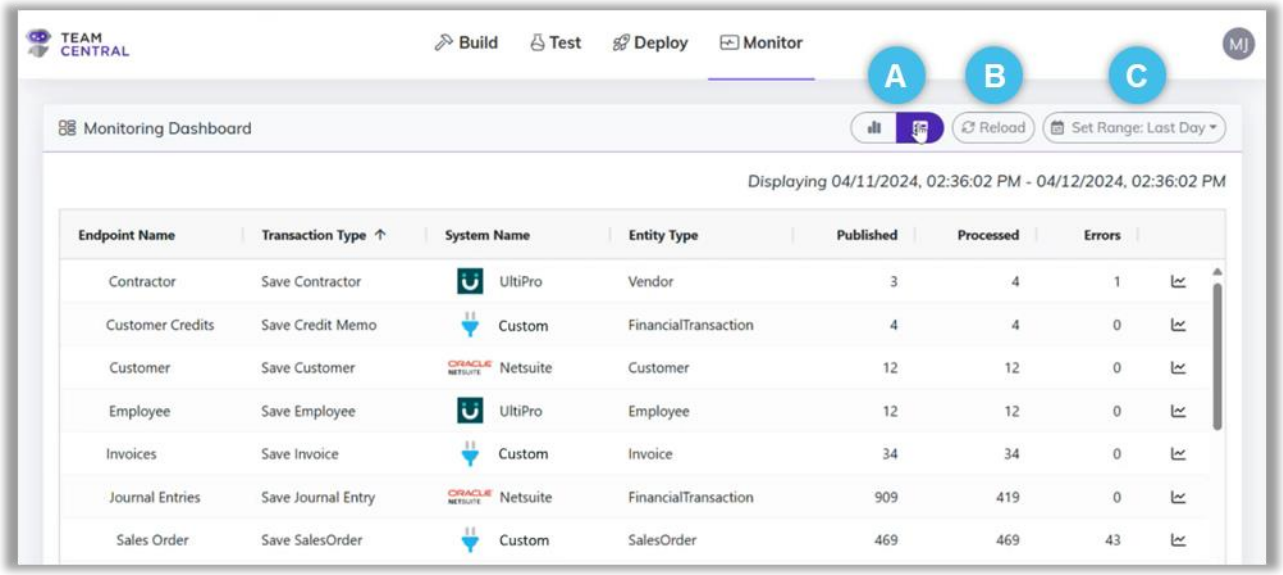
To:	Do This:
View the data in graph format	• Select the **graph** icon (A). 
View the data in table format	• Select the **table** icon (A). 
Reload	• Select **Reload** (B) to refresh the data.
Set a specific date range	• Select **Set Range** (C). • Select a pre-formulated **timeframe** (e.g. Last Hour) or select **Custom** to apply a specific date and time to view that data. • Select **Apply**. **Note:** By default, the system will show data for the **Last Day**. 

Graph View Dashboard



Status	Definition
Monitoring Dashboard	• Visual depiction of each systems' errors and published messages for the given date range.
Total Messages	• A quantified depiction of the total published and successful messages, as well as errors. **Note:** Below the Total Messages box, you will see a visual representation of this data as well.

Table View Dashboard



NOTES:

Select the **filter** icon or **sort** arrow in any column header to narrow your search. The filters will vary, based on that column's data. You can apply filters to one or multiple of the columns.

Select the **graph** icon in the final column of any row to view that specific Endpoint's data in a visual format.

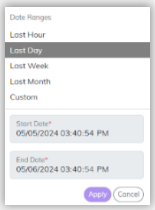
Column	Description
Endpoint Name	• The data is being published to this Data Hub Endpoint, which you established when creating the Data Hub.
Transaction Type	• Describes the logical grouping of the type of data event being transcribed, e.g. Save Employee Data, Save Invoice, etc. • This is the trigger point for data publishing; while the Entity Type refers to the type of data that it's tied to on the Common Model.
System Name	• The Connector system publishing that data within that Data Hub Endpoint.
Entity Type	• The type of data represented in the payload of the message, referencing the type of data that it's tied to on the Common Model. • This field directly ties back to the Schema Definition.
Published	• The number of data messages published within the set date range.
Processed	• The number of data messages processed within the set date range.
Errors	• The number of error messages that occurred within the set date range.

View Error Logs

NOTE: Error data can be found via several features, such as **Monitor** > **Error Logs** and **Monitor** > **Message Search**, as well as the **Actions** > **Message Search** and **Actions** > **Message Details** for each instance (row) within the Error Logs feature.

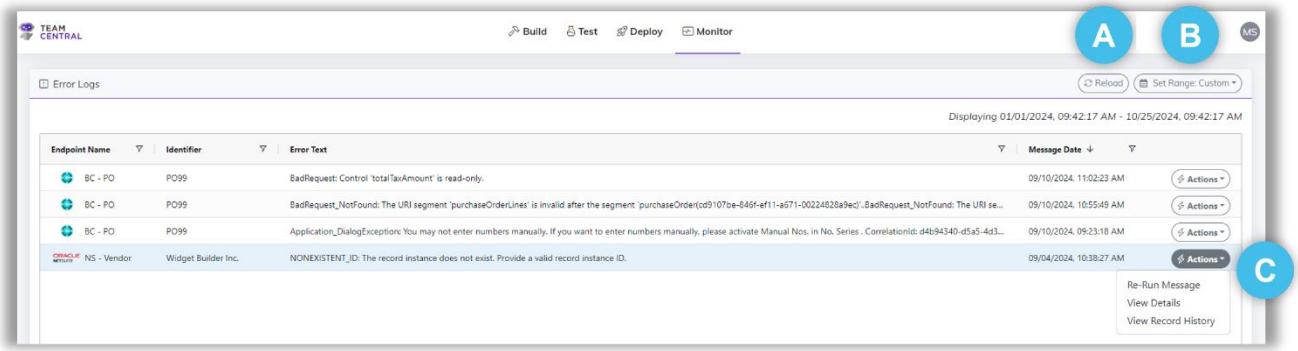
For those with advanced skills, error messages can also be found within the code view.

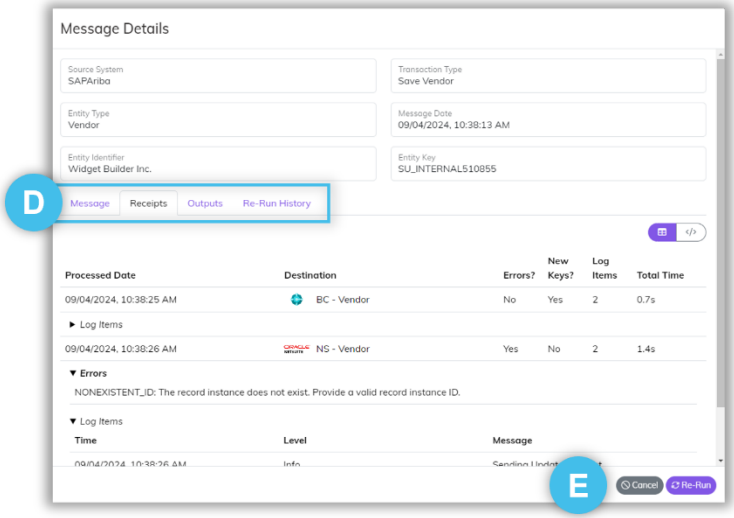
1. Select **Monitor** > **Error Logs**.
2. Then, use the below table to determine the next steps.

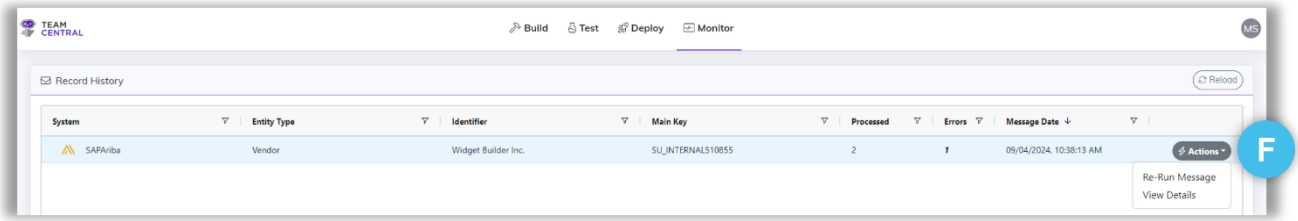
To:	Do This:
View the data	• All transaction data will be displayed in the table.
Refresh the data	• Select **Reload** (A).
View a specific time range	• Select **Set Range** (B). • Select a pre-formulated **timeframe** (e.g. Last Hour) or select **Custom** to apply a specific date and time to view that data. • Select **Apply**. **Note:** By default, the system will show data for the Last Day. 
Re-run the message	• Select **Actions** > **Re-Run Message** (C) in the appropriate row. **Note:** This option should be used only if you understand the error and believe it has already been fixed in the triggering system.

View message details	• Select **Actions** > **View Details** (C) in the appropriate row.
	• A Message Details window will appear; select a **tab** (D) to view additional information:
	  ○ **Message:** View the details in code view.
	  ○ **Receipts:** View all subscribers that picked up that message and tried to process it, as well as Error Log data.
	  ○ **Outputs:** Data will be populated if the subscriber endpoint has an entity type of "File," and will include a link to download the file that was created when the integration was triggered to execute.
	  ○ **Re-Run History:** View an audit of the previous re-runs, including the Date, By, and any Comments associated with that re-run.
	• Select **Cancel** (E) to close the pop-up window and return to the previous screen.
	
	Note: It is advised to view the message details, which provides an opportunity to analyze the error details and formulate a fix; however, as needed, you can also select to **Re-Run** the message here.

View record history	• Select **Actions** > **View Record History** (C) in the appropriate row. • The **record history** (F) for that error will open in a new browser tab, displaying all messages associated with that record, including all activity and errors with timestamps. ○ Select **Actions** > **Re-Run Message** or **Actions** > **View Details** from this screen to execute those actions. **Note:** This is a good way to view if the error has already been resolved. For example, if an error happened when syncing a sales order at 8 PM, but was successful when it ran again at 9 PM, both messages would appear in the history.

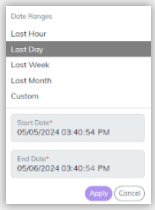






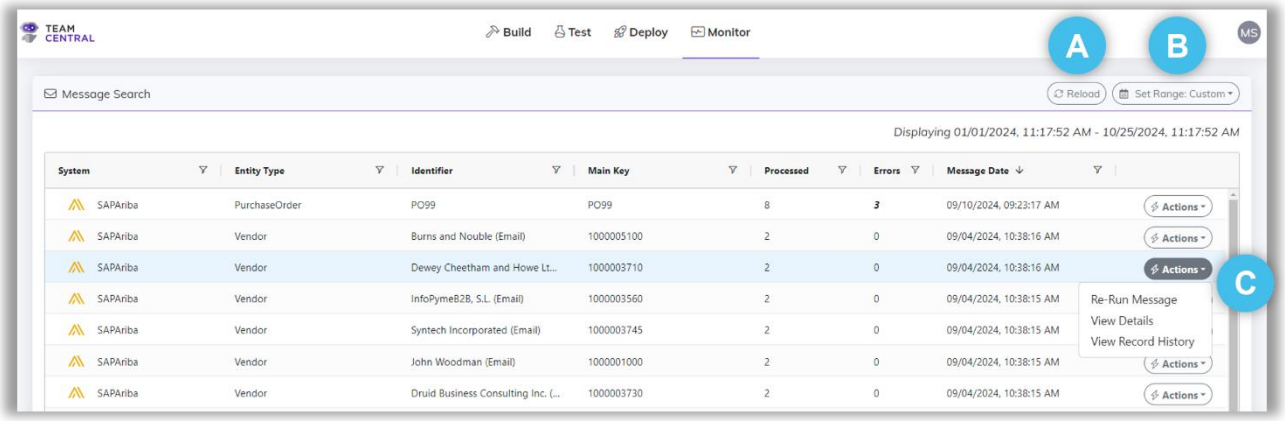
View Message Search

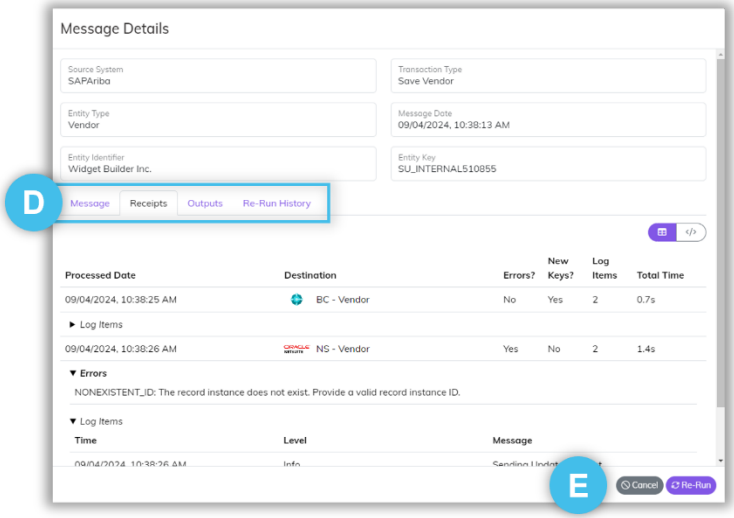
1. Select **Monitor** > **Message Search**.
2. Then, use the below table to determine the next steps.

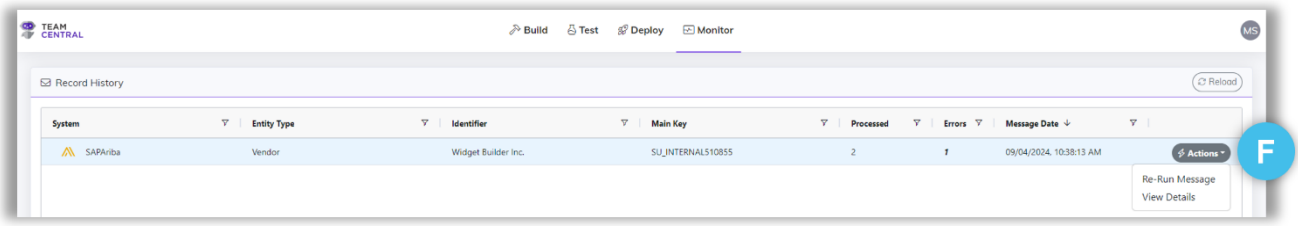
To:	Do This:
View the data	• All transaction data will be displayed in the table.
Refresh the data	• Select **Reload** (A).
View a specific time range	• Select **Set Range** (B). • Select a pre-formulated **timeframe** (e.g. Last Hour) or select **Custom** to apply a specific date and time to view that data. • Select **Apply**. **Note:** By default, the system will show data for the Last Day. 
Re-run the message	• Select **Actions** > **Re-Run Message** (C) in the appropriate row. **Note:** This option should be used only if you understand the error and believe it has already been fixed in the triggering system.

View message details	• Select **Actions** > **View Details** (C) in the appropriate row.
	• A Message Details window will appear; select a **tab** (D) to view additional information:
	  ○ **Message:** View the details in code view.
	  ○ **Receipts:** View all subscribers that picked up that message and tried to process it, as well as Error Log data.
	  ○ **Outputs:** Data will be populated if the subscriber endpoint has an entity type of "File," and will include a link to download the file that was created when the integration was triggered to execute.
	  ○ **Re-Run History:** View an audit of the previous re-runs, including the Date, By, and any Comments associated with that re-run.
	• Select **Cancel** (E) to close the pop-up window and return to the previous screen.
	
	Note: It is advised to view the message details, which provides an opportunity to analyze the error details and formulate a fix; however, as needed, you can also select to **Re-Run** the message here.

View record history	• Select **Actions** > **View Record History** (C) in the appropriate row. • The **record history** (F) for that error will open in a new browser tab, displaying all messages associated with that record, including all activity and errors with timestamps. ○ Select **Actions** > **Re-Run Message** or **Actions** > **View Details** from this screen to execute those actions. **Note:** This is a good way to view if the error has already been resolved. For example, if an error happened when syncing a sales order at 8 PM, but was successful when it ran again at 9 PM, both messages would appear in the history.

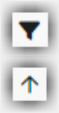






Message Search Reference Table

NOTE: Hover over and select the **filter** icon or **sort** arrow in any column header to narrow your search. The filters will vary, based on that column's data. You can apply filters to one or multiple of the columns, as needed.



Status	Definition
System	• The system that triggered the message.
Entity Type	• The type of data represented in the payload of the message.
Main Key	• The unique identifier for that record in the system that triggered the message.
Processed	• The number of messages processed within the set date range.
Errors	• The number of errored messages within the set date range.
Message Date	• The date and time stamp that message occurred.

Example: If you find that the Main Key with the Message Search error shows 3186, and the System shows it was generated by NetSuite, then you can go to your NetSuite application to view record 3186 to gather additional information.

Central Common Model Definitions

Term	Definition	Example
Asset	A resource owned by the company that has economic value, such as equipment, vehicles, or intellectual property.	A company car used by the sales team, or a software license purchased for office use.
BOM Revision	A version of a Bill of Materials (BOM) that reflects changes or updates to the components required to manufacture a product.	The latest version of a BOM to include a new type of screw in the assembly of a product.
Company	A legal entity or business organization that a company may have a relationship with, such as a supplier, vendor, customer, or partner.	"ABC Manufacturing Inc." is a company that is a current customer.
Expense Entry	A record of costs incurred by the company, typically entered for reimbursement or accounting purposes.	A $50 expense entry for office supplies purchased by an employee.
File	A digital document or attachment stored within a system, often linked to specific records.	A PDF of a signed contract uploaded to a vendor's profile.
Financial Account	An account used to record financial transactions, such as bank accounts, cash accounts, or credit card accounts.	The company's checking account at a local bank.
Financial Transaction	A record of any exchange or transfer of financial value within the company.	A $500 payment to a supplier for raw materials.
Inventory Transaction	A record of movements or changes in inventory levels, such as stock additions or withdrawals.	Receiving 100 units of a product into the warehouse.
Invoice	A document issued to customers or from vendors detailing goods or services provided and the amount owed.	An invoice sent to a customer for a recent order of 50 products.
Lead	A potential customer or client that has shown interest in the company’s products or services.	A contact form submission from someone interested in your software.
Opportunity	A potential sales deal or business venture that the company is pursuing.	A negotiation with a prospective client to close a deal worth $100,000.
Organizational Unit	A subdivision of the company, such as a department or team, that performs specific functions.	The Marketing Department or the Sales Team.

Package	A grouping of items or components that are shipped or handled together.	A package containing 5 different products shipped to a customer.
Person	An individual associated with the company, or with a customer or vendor of the company, such as an employee, contractor, or contact.	John Doe, a sales representative at the company.
Product	An item or service offered by the company for sale to customers.	A new laptop model available for purchase.
Production Planning Schedule	A timeline or plan outlining when and how products will be manufactured.	A weekly schedule detailing the production of 500 units of a specific product.
Project	A temporary endeavor undertaken by the company to create a unique product, service, or result.	A project to develop a new software module for a client.
Purchase Order	A document issued by the company to a supplier, requesting the purchase of goods or services.	A purchase order for 200 units of raw materials.
Sales Order	A document confirming a customer's order for products or services.	A sales order for 100 units of a product placed by a customer.
Shipment	The process or record of sending goods from the company to a customer or another location.	A shipment of products sent to a retailer.
Time Entry	A record of hours worked by an employee or contractor, often used for payroll or billing purposes.	A time entry for 8 hours worked on a specific project.
Vendor Bill	An invoice received from a supplier for goods or services provided to the company.	A bill from a supplier for raw materials delivered last month.
Work Order	An instruction or authorization to perform work, often related to manufacturing or maintenance tasks.	A work order to repair a piece of machinery on the production floor.

Appendix A: Connectors Catalog

NOTE: The below table provides links to common third-party software selector authentication documentation, all common to Central data integrations. TeamCentral is not responsible for the information provided within these materials (links).

NOTE: Most Connectors use one of the following query languages:

- For JSON, you'll be looking for the JSONPath documentation.
- For XML, you'll be looking for the XPath documentation.

Connector	Link to Selector Authentication Information
ADP	ADP API Explorer
Active Directory	AD API Documentation
Acumatica	Acumatica API Documentation
Autodesk	Autodesk API Documentation
Benchmark	Benchmark Email RESTful API v3.0
BigCommerce	API Reference Documentation
BigQuery	BigQuery API
Degreed	Degreed API Reference Documentation
Dynamics 365 (Business Central)	API v2.0 for BC Documentation
Dynamics 365 (Customer)	Dynamics 365 Customer Engagement REST API Web API
FTP Server	Connect to FTP Server from Workflows in Azure Logic Apps
HubSpot	HubSpot API Reference Documentation
Hy-Tek	API documentation is not publicly available; contact vendor for support.
JobBOSS	API documentation is not publicly available; contact vendor for support.
Magaya	Magaya API Documentation
MiniBC	API and Webhook Documentation
NetSuite	NS API Reference
Office 365	Office 365 Management API Overview
Onfleet	Onfleet API Documentation
OpenAir	NS OpenAir API Documentation

Payload	Payload APIs
Procore	API Resource Guide - Overview
RedSky Mobility	API documentation is not publicly available; contact vendor for support.
Rent Manager	Rent Manager 12 Web API Overview
SAP Ariba	SAP Ariba API General Documentation
SAP Ariba (SOAP)	Main Index of SOAP Documentation
SQL Server Database	API documentation is not publicly available; contact vendor for support.
Salesforce	SF API General Documentation
Sovos	API General Documentation
Stedi	API Reference