

tableBASE RESTConnector

Release 7.0.3

Copyright © 2026 DataKinetics Ltd.

Document Number: TBM011-R703v1.0

Publication History:

tableBASE Release 7.0.3 Version 1.0 - July 2026

This guide is copyrighted and all rights are reserved. This document may not, in whole or in part, be copied, photocopied, reproduced, translated or reduced to any electronic medium or machine-readable form without the prior written consent of DataKinetics Ltd.

Information in this guide is subject to change without notice and does not represent a commitment on the part of the vendor. The software described in this guide is furnished under a license agreement and may be used or copied only in accordance with the terms of the agreement.

tableBASE, tablesONLINE, VTS Manager and VTS Gate are registered trademarks of DataKinetics Ltd. The names of other products or companies may be trademarks or registered trademarks of their respective companies.

DataKinetics Technical Support: 1-613-523-5588

Email: tableBASE@dkl.com

DataKinetics Ltd.

110 Didsbury Rd.

Kanata, ON

Canada K2T 0C2

<http://www.dkl.com>

Telephone: (613) 523-5500
1-800-267-0730 (toll free in the US and Canada)

Contents

List of Figures.....	v
List of Tables.....	ix
Preface.....	xi
Audience for this Guide.....	xi
What is Covered in this Guide.....	xi
Naming protocol.....	xi
Glossary.....	xi
Conventions used in this guide.....	xiv
Additional tableBASE references.....	xv
Customer Support.....	xv
Training and Professional Services.....	xvi
About DataKinetics.....	xvi
 Chapter 1: Introduction.....	 1
REST Architecture.....	2
Accessing tableBASE Tables.....	2
Challenges for Enterprise IT.....	2
RESTConnector for tableBASE.....	3
Solutions offered by RESTConnector.....	3
No-Code Approach.....	3
In memory.....	3
 Chapter 2: Getting Started with tableBASE RESTConnector.....	 5
Design tableBASE tables and Load Data into the Library.....	5
Opening a Table in a TSR or VTS-TSR.....	6
Creating an Alternate Index.....	7
Opening a table in a VTS-TSR.....	7
Customize the Copybook.....	8
Customizing a Copybook to Retrieve a Single Record.....	8
Customizing the Copybook to Retrieve Multiple Records.....	9
Customizing the Copybook to Retrieve Multiple Hexadecimal Records.....	9
API Input Request Fields.....	10
API Output Response fields.....	11
 Chapter 3: Designing APIs for Single Record Retrieval.....	 13
Prerequisites.....	13
Set Perspective.....	13
Create a New z/OS Connect EE Service Project.....	14
Create a New Service Project.....	14

Create New Service Interface.....	16
Deploy the Service to the Server.....	20
Create a New API Project.....	21
Import the Service Archive.....	25
Mapping the Request.....	27
Mapping the Response.....	34
Deploy API to Server.....	36
Chapter 4: Designing APIs for Multiple Record Retrieval.....	37
Prerequisites for Multiple Responses.....	37
Create a New Service Project for Multiple Records.....	37
Create a New Service Request for Multiple Records.....	39
Create new Service Response for Multiple Records.....	42
Deploy the Service Project.....	49
Create a New API Project for Multiple Records.....	50
Mapping the GET Method.....	54
Import the Service to the GET Method.....	54
Mapping the GET Request for Multiple Responses.....	56
Mapping the GET Response for Multiple Record Retrieval.....	65
Deploy API to server.....	68
Create a new Service Project for Multiple Records with Hexadecimal response.....	69
Create a New Service Interface for Multiple Records.....	69
Create New Service Response for Multiple Records with Hexadecimal Response.....	70
Deploy The Service Project for Multiple Records with Hexadecimal Response.....	70
New API Project for Multiple Records with Hexadecimal Response.....	70
Mapping the GET Method.....	70
Map the Service to the GET Method.....	71
Map the Get Request for Multi Responses.....	71
Mapping the GET Response for Multiple Record Hexadecimal Retrieval.....	71
Chapter 5: Testing the RESTConnector.....	73
Testing Single Response.....	73
Testing Multiple Response.....	75
Appendix A: Error Messages.....	79
How to Understand the Error Conditions.....	79
CICS Errors.....	79
tableBASE Errors.....	79
API Errors.....	79
Error Codes.....	80

List of Figures

Figure 1-1: RESTConnector Flow Diagram.....	1
Figure 3-1: Open Perspective Window.....	14
Figure 3-2: New Project Window for Service Project.....	15
Figure 3-3: Service Project Editor.....	16
Figure 3-4: Create Service Interface Button.....	16
Figure 3-5: Create Service Interface window.....	17
Figure 3-6: CATLG Edit Container.....	17
Figure 3-7: Edit Container Window.....	17
Figure 3-8: Import data structure.....	18
Figure 3-9: Import Window.....	18
Figure 3-10: Import Application Data Structure Source window.....	19
Figure 3-11: Import Window populated.....	19
Figure 3-12: DK1APIC1-Con Expanded.....	20
Figure 3-13: Deploy Service.....	21
Figure 3-14: Deploy Service to z/OS Connect EE Server Result.....	21
Figure 3-15: CATLG z/OS Connect EE Servers.....	21
Figure 3-16: New API Project.....	22
Figure 3-17: API Editor with Four Methods before removal.....	22
Figure 3-18: Path equals itemsID.....	23
Figure 3-19: Export Service Package Window.....	23
Figure 3-20: Select a Workspace Folder.....	24
Figure 3-21: CATLGAPI Export Service Package window.....	24
Figure 3-22: GET Method.....	25
Figure 3-23: Select zOS Connect EE Service window.....	25
Figure 3-24: Import z/OS Connect EE Services window.....	26

Figure 3-25: Import Services window.....	26
Figure 3-26: GET Service mapped.....	27
Figure 3-27: Open Both Mappings.....	27
Figure 3-28: Request and Response Tabs.....	28
Figure 3-29: Add Assign Transform.....	29
Figure 3-30: Assign IN_API_KEY.....	30
Figure 3-31: Assign Table Name.....	31
Figure 3-32: Assign VTS Name.....	32
Figure 3-33: Add Remove Transform.....	33
Figure 3-34: Request Mapping.....	34
Figure 3-35: Response Mapping.....	35
Figure 3-36: Deploy API window.....	36
Figure 3-37: Deploy API to zOS Connect EE Server Result window.....	36
Figure 3-38: catlgapi started.....	36
Figure 4-1: Create a new Service Project.....	38
Figure 4-2: CICSConn Configuration.....	39
Figure 4-3: Service Project Editor Definition tab.....	39
Figure 4-4: Define the Service Request.....	40
Figure 4-5: Create Service Interface Window.....	40
Figure 4-6: CATLG Service Interface editor.....	40
Figure 4-7: Edit Container Window.....	41
Figure 4-8: CATLG Import Data Structure.....	41
Figure 4-9: Source PDS Import Request.....	41
Figure 4-10: CATLG Import Application Data Structure Source window.....	42
Figure 4-11: CATLG Request Service Interface.....	42
Figure 4-12: Create Service Interface Button.....	43
Figure 4-13: Create Service Interface Window.....	43
Figure 4-14: Edit Container.....	43

Figure 4-15: Edit Container Window.....	44
Figure 4-16: Import Structure.....	44
Figure 4-17: Import Window for Source PDS.....	45
Figure 4-18: Import Application Data Structure Source Window.....	46
Figure 4-19: Disable OUT_RECS_THIS_CALL.....	47
Figure 4-20: Array Definition.....	48
Figure 4-21: OUT_RECS_THIS_CALL disabled.....	49
Figure 4-22: Response Service Interface.....	49
Figure 4-23: CATLGM Deploy Service.....	50
Figure 4-24: Deploy Server to z/OS Connect EE Server Result Window.....	50
Figure 4-25: CATLGM Deployed.....	50
Figure 4-26: New API Project.....	51
Figure 4-27: catlgmapi Editor.....	52
Figure 4-28: GET Method.....	52
Figure 4-29: Export Service Package.....	53
Figure 4-30: Select a Workspace Folder Window.....	53
Figure 4-31: Completed Export Service Package.....	54
Figure 4-32: Service to GET.....	54
Figure 4-33: Select z/OS Connect EE Service Window.....	55
Figure 4-34: Import z/OS Connect EE Services Window.....	55
Figure 4-35: Import Services.....	56
Figure 4-36: CATLGM Open both mappings.....	56
Figure 4-37: Request and Response Mapping.....	57
Figure 4-38: Add Assign Transform.....	58
Figure 4-39: Assign IN_API_KEY.....	59
Figure 4-40: Assign IN_TABLE_NAME.....	60
Figure 4-41: Assign VTS Name.....	61
Figure 4-42: Request Add Remove Transform.....	62

Figure 4-43: Path Parameter item-desc.....	63
Figure 4-44: LIMIT Query Parameter.....	64
Figure 4-45: OFFSET Query Parameter.....	64
Figure 4-46: Query Parameters assigned.....	65
Figure 4-47: Response Mapping the Copybook.....	66
Figure 4-48: Response Add HTTP Header.....	67
Figure 4-49: Add HTTP Header Window.....	67
Figure 4-50: OUT_TOTAL_RECS mapped to HTTP Header.....	68
Figure 4-51: Deploy CATLGMAPI.....	68
Figure 4-52: Deploy API to zOS Connect EE Server Result.....	69
Figure 4-53: catlgmapi started.....	69
Figure 5-1: REST API GET Method.....	73
Figure 5-2: GET button response operation.....	74
Figure 5-3: Successful Output Response.....	75
Figure 5-4: Multiple Testing Response Successful.....	76
Figure 5-5: Multiple Testing for Hexadecimal Response Successful.....	77

List of Tables

Table 2-1: API Type..... 8

Table 2-2: API Input Request Fields..... 10

Table 2-3: API Output Response Fields..... 11

Table A-1: Error Messages..... 80

Preface

This guide provides an overview of the RESTConnector of tableBASE.

Audience for this Guide

This guide and the *tableBASE Release Notes* are intended for use by system installers and/or system administrators responsible for installing the tableBASE Version 7.0.3 software. The person installing tableBASE should be familiar with a programming language like COBOL, PL/I, Assembler or C, compile procedures, the linkage editor, and MVS JCL.

If you are installing a new *version* of tableBASE or a new *release* for the first time, for example, moving from Version 6 to Version 7 or installing Release 7.0.3 for the first time, you should continue following the steps and procedures in this guide. If you are unsure about the distinction we are making here between the terms *version* and *release*, see the [Conventions used in this guide](#) on page xiv below for more detail.

If you are upgrading to a new *release or modification level* of tableBASE from a previous *release* (without adding a new product or component), you should consult the relevant installation instructions in the *tableBASE Release Notes*, which will explain the most efficient way to accomplish your upgrade.

What is Covered in this Guide

This guide contains the installation procedure for the tableBASE base product (batch interface) as well as the procedures for the installation of the optional CICS and IMS TM interfaces, the VTS (Virtual Table Share) product, the VTS Gate product, and the tablesONLINE products for CICS and ISPF. Also covered in this guide are parameter settings, parameter delivered defaults, and overrides.

Naming protocol

Version 6 featured the new tableBASE naming protocol which remains with Version 7.0.3. All tableBASE executables begin with DK1 for easy identification, a prefix that has been reserved for exclusive use with IBM.

Aliases are retained so that no changes are required to your existing applications.

Modules received for the installation should be used to entirely replace earlier Version 6 modules.

Glossary

The following terms are used throughout this guide:

Alias name	An alternate name for a VTS-TSR that can be created, assigned, and used in lieu of the name assigned at VTS-TSR definition. This is a VTSManger feature. See below for VTSManger and VTS-TSR definition.
Alternate Index	An Alternate Index is an Index that may be defined for a Data Table. The Alternate Index has an Alternate Index definition (ALT-DEFINITION) that defines the key, organization, and search order. Alternate Indexes are optional, and there is no limit to the number of Alternate Indexes a Data Table may have.
Alternate Index View	A tablesONLINE Alternate Index View is identical to a View but applies to a Data Table when access is through an Alternate Index.
Data Table	A Data Table is the actual raw data. Each Data Table has a table definition (DT-BLOCK) that is used to generate the Index for the Data Table.
DB2 SPAS, WLM-managed DB2 SPAS	DB2 Stored Procedure Address Space managed by WLM.
Delivered defaults	The defaults that are delivered with the product. Also known as <i>factory defaults</i> .
ETL	The Extract Transform and Load Batch utility program (DK1ETL) Extracts data from the original DB2 or VSAM source, transforms it according to control card criteria, and loads it into tableBASE libraries and tables.
Index	An Index is defined for each Data Table. A Data Table Index is generated dynamically when a table is opened or defined based on the information in the table definition (DT-BLOCK).
Installation defaults	The defaults set at installation time by an administrator, which may or may not be the same as the delivered defaults. Defined using the TBOPTGEN file. (These defaults may be overridden by an individual application using the TBOPT file.)
LDS	Linear DataSet used for VTSManger.
Linked Table	A linked table (also known as a remote table) is created when a user issues a command to open a table that is already open in a VTS-TSR specified in the LIB-LIST. The table entry in the local TSR is linked to the existing open table in the VTS-TSR. No updates are allowed to a linked table.

Multitasking Batch	An MVS region that implements multitasking by attaching multiple Task Control Blocks (TCBs). This can include a batch job that attaches several subtasks or a transaction processing region like DB2 stored procedures that implements multitasking through multiple TCBs.
PC Server	The PC Server allows tableBASE program calls (PCs) to be available for tableBASE regions. The PC functionality supported by the PC Server provides the functions needed for tableBASE multi-tasking, for CICS thread-safe support, for accessing a VTS-TSR, and for compatibility with DB2 stored procedures, as well as other special functions that are required by tableBASE.
SAF interface	System Authorization Facility interface that controls resource access. Tools like ACF2®, RACF®, and Top Secret® fall into this category.
SPACEMAP	The name of the memory allocator used and manage tableBASE TSRs.
Table Expansion	Dynamic allocation of space for tables in the TSR when the current space allocated becomes insufficient.
Temporary Table	A temporary table exists only within a TSR, and is created by the DT command (or IA). It is never stored in a library. A temporary table can be distinguished from a library table using the GD command output—if found, a temporary table will show no dataset name.
TSR, Local TSR	Table Space Region. A data space of up to 2G is used by tableBASE to house tables. The data space is owned by an application in the associated address space. The application uses tableBASE to access data within the tables.
TSR of record	If there is no VTS name in the TB-SUBSYSTEM field of the TB-PARM when a command is being executed, the TSR of record is the local TSR, otherwise it is the VTS-TSR that has its name in the TB-PARM.
View	A tablesONLINE View provides the field, edit and display attributes for a Data Table with its Index. Previous to Version 5, the View was referred to as a Field Definition Table (FDT).
VTS, DataKinetics VTS	Virtual Table Share. The term "VTS" refers to the DataKinetics VTS product, which permits TSRs to be shared among applications. These shared TSRs are called VTS-TSRs.

VTS-TSR, Shared TSR	A Virtual Table Share (VTS) Table Space Region (TSR) is a shared TSR, and resides in a shared data space. Applications can access tables within a VTS-TSR, and use the information as if it were within their local TSRs.
VTS Agent	The DK1VAGNT program, which initializes VTS-TSRs in tableBASE, and then sits idle until the VTS-TSR is to be terminated.
VTSGate	The product that provides enhanced security for the shared VTS-TSR capability.
VTSManger	A DataKinetics Ltd. product that extends the functionality of the VTS shared TSR capabilities.
VTS switch	The action of switching an alias name from one VTS-TSR to another. This is a feature of VTSManger.

Conventions used in this guide

This guide uses certain conventions to differentiate code and typed commands, to display the names of parameters, and to describe specific version and release levels of the tableBASE product.

Convention	Description
code examples and commands	Code examples and commands appear in this type of font: <code>this is an example of the font.</code>
MAXNMTAB	Names of parameters appear in upper case simply for ease of reading; actual case used is upper or lower or a mixture.
Version	Following IBM standards, the term <i>version</i> refers to a generation of a software product that has significant new code or new functionality. <i>Version</i> is a more general term than <i>release</i> . For example, <i>Version 6</i> includes <i>Release 6.1</i> and <i>Release 6.2</i> , and is equivalent to <i>Release 6.x</i> .
Release	Following IBM standards, the term <i>release</i> refers to a program or set of programs which represent a specific revision to the base version of a software product. For example, <i>Release 6.0</i> is a term that is used to identify the first release of <i>Version 6</i> . Subsequent releases made available under the Version 6 umbrella, such as <i>Release 6.1</i> , will provide additional revisions to the base product.

Convention	Description
Modification Level	Following IBM standards, the term <i>modification level</i> refers to the application of specific program enhancements and error corrections to the release of a software product. For example, <i>Release 6.0.3</i> is at <i>modification level 3</i> , and <i>Release 6.1.0</i> is at <i>modification level 0</i> .
MVS	MVS is a generic term which is used when referring to z/OS and other related IBM operating systems.

Additional tableBASE references

This guide is one of a series that describe tableBASE and tablesONLINE:

- *tableBASE Release Notes*
- *tableBASE Installation Guide*
- *tableBASE Concepts and Facilities Guide*
- *tableBASE Batch Utilities Guide*
- *tableBASE Administration Guide*
- *tableBASE Programming Guide*
- *tableBASE Quick Reference Guide*
- *tablesONLINE/CICS User's Guide*
- *tablesONLINE/ISPF User's Guide*
- *tableBASE Library Bridge Manual, Release 5.B*

Customer Support

Customers with maintenance agreements can obtain 24-hour hotline support through a direct line to our Technical Support staff (+1-613-523-5588).

The DataKinetics customer support Service Level Agreement lists the guaranteed response times for each problem severity level. High priority critical and serious problems receive immediate attention from our Technical Support staff.

Before calling the DataKinetics Technical Support staff, be sure to consult the support section of our web site (www.dkl.com) for all the latest information on our technical support protocol. Look in the "Customer Support Portal" for our "FAQs" which provide answers to frequently asked questions posed by our users.

The "Customer Login" link supplied on our web site navigates to a special support section which is available to customers who have maintenance agreements with DataKinetics Ltd. A username and password are needed to access this section of the web site—consult your tableBASE administrator, or DataKinetics customer support, for this information.

Training and Professional Services

DataKinetics Ltd. is proud to be able to offer our customers the very best in training and technical consulting services. See the "Services" section of our web site for all the latest information regarding our offerings in this area.

About DataKinetics

DataKinetics®, the leader in enterprise IT optimization, enables the top data centers in the world to overcome today's mainframe throughput, performance, capacity and interconnectivity challenges. In business for over 40 years, DataKinetics products are powering nearly a billion mission-critical transactions per second every day for the world's largest banking, insurance, credit card, brokerage house and retail organizations.

Chapter 1

Introduction

This chapter introduces tableBASE RESTConnector, explains its relationship to tableBASE, and describes how to configure and deploy APIs that interact with tableBASE data.

A full featured table manager, tableBASE is specifically designed to enable the rapid development of applications that require many table reads. It manages tables in memory and allows multiple users to concurrently read, modify, and permanently store data.

tableBASE is a complete infrastructure for defining, building, maintaining, controlling, and using table data. Application performance and productivity can be improved by placing tables with these types of data into memory.

It is designed to work only with memory-resident tables and to help improve performance and productivity issues relating to tables.

The RESTConnector is an enhancement to tableBASE that opens the data stored in tableBASE as RESTful APIs. The data can be consumed and utilized by more modern applications such as mobile and cloud-based applications.

RESTConnector is designed as a no-code approach to deploy APIs on IBM® System Z. The only change an API designer must do in the back-end system is customize the copybook which is later uploaded to IBM® z/OS Connect to create a Service Archive File (.sar). A detailed step-by-step approach on how to configure and deploy APIs can be found in this guide.

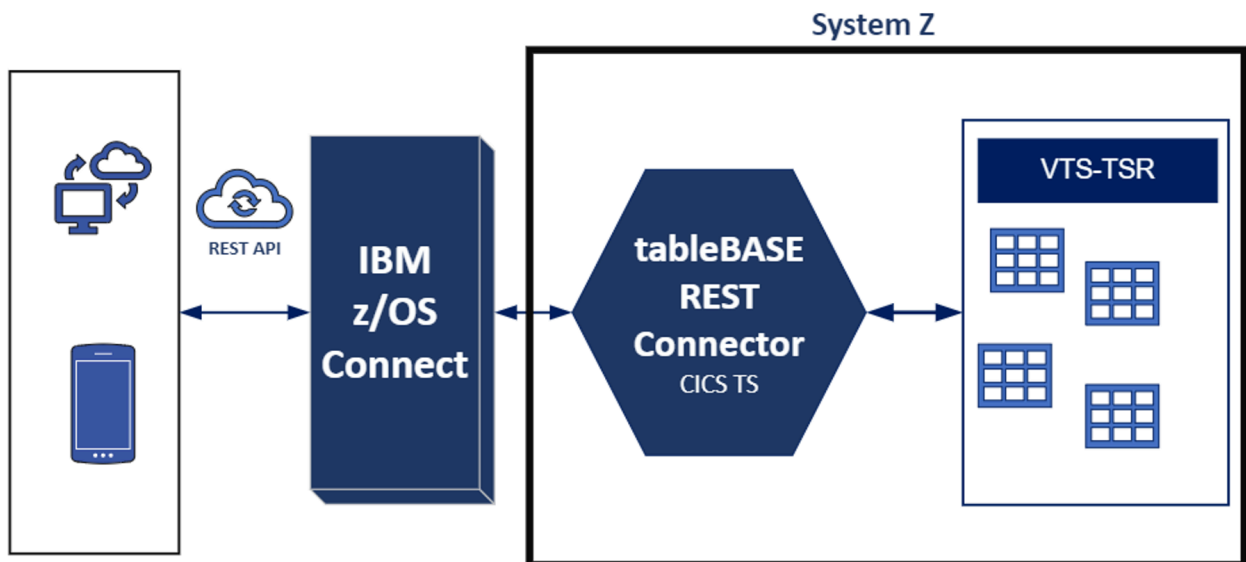


Figure 1-1: RESTConnector Flow Diagram

A VTS-TSR containing tableBASE tables processed through the RESTConnector using IBM z/OS Connect creates a REST API. More information can be found at the topic about [z/OS Connect Open API 2](#) on IBM's website.

REST Architecture

Representational State Transfer (REST) is an architectural style designed for scalability of interactions between the components, uniform interfaces, independent deployment components across platforms, enforce security, and encapsulate legacy systems.

REST has been implemented throughout the software industry and is widely accepted to create stateless, reliable APIs across multiple platforms. REST APIs are typically based on HTTP methods to access resources via URL-encoded parameters and use JSON to transmit data.

Accessing tableBASE Tables

Methods to access tableBASE tables include:

DK1TDRV - a program intended for application developers to test tableBASE calls interactively. It can be used for testing command sequences before committing the sequence to program logic.

DK1TDRV can operate as a complete interactive table access facility. For more information refer to the section about DK1TDRV in the *tableBASE Programmer's Guide*.

DK1TCALL - an Application Program Interface (API) which issues tableBASE calls within batch programs.

DK1TEXEC - a batch table manipulation utility. Access the table with batch application software or DK1TEXEC. For more information about this subject, refer to the *tableBASE Programmer's Guide* and the *tableBASE Batch Utilities Guide*.

DK1TPTBL - a batch table printing utility.

DK1TVWPR - a batch table definition print utility.

Challenges for Enterprise IT

As organizations move toward hybrid cloud infrastructure, there is an increasing need for applications to interact more securely and effectively across multiple platforms including Cloud and On-Premises platforms. IT Professionals are faced with one or more of the following related challenges:

- Ability to quickly deploy APIs on System Z.
- Challenges around implementation of Dev Ops practices to traditional z/OS applications.
- Data transformation challenges between z/OS and other non z/OS platforms.
- Looking to provide increased security for sensitive data.

RESTConnector for tableBASE

tableBASE uses proprietary high-performance software technology to manage mainframe-based in-memory table for use in transaction-intensive environments.

It improves mainframe performance by:

- making mainframe data more accessible and available
- reduces the need for upgrades
- greatly simplifies mainframe business rules processing.

RESTful APIs have established leading standards for new application development. z/OS Connect provides a structure enabling z/OS-based programs and data to engage in modern technology for mobile and cloud applications.

RESTConnector for tableBASE provides a no-code approach to deploy REST APIs to expose data stored in tableBASE local TSRs and VTS-TSRs.

Solutions offered by RESTConnector

RESTConnector is an enhancement to the base product tableBASE. RESTConnector acts as a middle component between tableBASE and z/OS Connect to transform a REST call into a native tableBASE API call that retrieves data from local TSRs and VTS-TSRs.

z/OS connect is not a mandatory requirement to use RESTConnector, however, z/OS Connect is recommended due to its ability to handle data transformation and easy integration capabilities with RESTConnector.

No-Code Approach

RESTConnector runs on System Z using a CICS Transaction Server. It is designed as a generic component that can handle any data layout. It requires a customized version of the COBOL copybook layout that matches the fields required in the API request and response.

This feature empowers the users with the ability to quickly deploy multiple APIs, reducing the effort and cost required to develop, test, and deploy code changes.

Additionally, when new fields are added or removed, only the copybook layout needs to be updated and uploaded into z/OS Connect. There is no need to ship any code changes to System Z.

In memory

RESTConnector is designed to retrieve data from a tableBASE VTS-TSR and a local TSR. A VTS-TSR is a shared dataspace managed by tableBASE. This feature enables RESTConnector to retrieve data from memory avoiding any disk I/O, providing an improved API response time.

Data from any existing source, such as flat files, VSAM, or DB2, can be transferred to tableBASE tables making them API-ready.

Chapter

2

Getting Started with tableBASE RESTConnector

This chapter explains how to use tableBASE RESTConnector interface to configure and deploy APIs to expose data stored in tableBASE.

Note: tableBASE RESTConnector interface is designed as a no-code approach. Product components are installed in the CICS region. Detailed information about the prerequisites and installation steps can be found in the *tableBASE Installation Guide*.

z/OS Connect is not a prerequisite to deploy the APIs using the RESTConnector interface. An example would be, native CICS Webservice facility can also be used to expose tableBASE data as CICS Webservices. z/OS Connect is recommended due to its rich features and ability to support OPEN API 2.0 and 3.0 standards. The *RESTConnector User Guide* provides detailed information on how to use IBM® Explorer for z/OS to configure APIs that interact with the tableBASE RESTConnector interface.

It is assumed that all the prerequisites tableBASE, tableBASE RESTConnector interface, and z/OS Connect or similar web services product are already installed and configured.

The following are the main steps required to get the data ready for the API.

1. Design tableBASE tables and load data into the tableBASE into the library.
2. Open the table in either a TSR or a VTS-TSR.
3. Generate and Customize the copybook.

Design tableBASE tables and Load Data into the Library

Designing your tableBASE table is a critical step in exposing API.

If the data is to be loaded exactly as it is from an existing data source, for example a physical sequential file or VSAM, tableBASE DK1TEXEC Utility can be used to define a tableBASE library, define a tableBASE table in the library, and load data into tables in the library. The provided sample JCL and control card can be used to get started with data that is available in a physical sequential file that is mapped by a DD name DATA. It will be loaded into a tableBASE library mapped by DD name MAINLIB as shown.

```
//DK1TEXEC EXEC PGM=DK1TEXEC
//STEPLIB DD DISP=SHR,DSN=your.tableBASE.load
//TBMSG DD SYSOUT=*
//TBRPT DD SYSOUT=*
//TBOPT DD *
TSRSIZE=2G
//DATA DD DISP=SHR,DSN=your.data.source
```

```
//MAINLIB DD DSN=your.tableBASE.library,
//          SPACE=(TRK,(100,199),RLSE),
//          DCB=(BLKSIZE=3120,LRECL=3120,RECFM=F,DSORG=PS),
//          DISP=(,CATLG)
//CNTLCARD DD *
DEFINE NEWLIB=MAINLIB;
DEFINE TBL=EXMPCAT EST=1000 RSZ=80 KSZ=4 KLOC=1 LIB=MAINLIB
MTHD=B ORG=S;
LOAD TBL=EXMPCAT FROM=DATA LIB=MAINLIB;
/*
```

Note: In this example, EXMPCAT is the name of a tableBASE table to store sample catalog manager application data which has key size of 4 bytes starting at key location 1. The tableBASE table is designed as a Sequential organization (ORG=S) that would use Binary search method (MTHD=B).

Detailed explanation of DK1TEXEC utility and its corresponding control cards can be found in the *tableBASE Batch Utilities Guide* in Chapter 2, DK1TEXEC Commands as well as Chapter 3, Using DK1TEXEC.

It is important to note how a tableBASE table is designed. APIs exposed via this interface directly correspond to how the data is organized in tableBASE table.

Key size and key location in the DEFINE TBL command play a critical role in how the API retrieves the data via a RESTful API HTTP query parameter.

If the HTTP GET request retrieves multiple records, an alternate index may be required on an alternate key. Details about how to create an alternate index can be found in the *tableBASE Programming guide*.

Opening a Table in a TSR or VTS-TSR

tableBASE uses dataspace to open tables in memory. The process of loading tables from a tableBASE library to dataspace is called opening the tables. These dataspace managed by tableBASE are called TSRs.

tableBASE uses two types of TSRs:

A **local TSR** is owned by an address space. A local TSR is owned by the respective address spaces that create them, example: Batch, CICS, etc. When the address space ends, the data loaded into local TSRs will be closed and removed. If using a local TSR, ensure the CICS Region uses the correct tableBASE library containing the data table.

A **VTS-TSR** is owned by the VTS Agent address space. This is a shared dataspace that can be used to share data across multiple address spaces including Batch and CICS regions.

tableBASE RESTConnector can be used to expose data from both a local TSR and VTS-TSR. VTS-TSRs are recommended because of their ability to share data across multiple address spaces.

Note: For the purposes of this guide a VTS-TSR will be used throughout the document.

Creating an Alternate Index

In the example provided, VTS2 is the name of the VTS-TSR the table EXMPCAT is loaded into. To access multiple records, an alternate index may need to be created on a field that will act as an alternate key.

In the EXMPCAT table, the "Product Description" field meets this criteria. EXMPCATA is an alternate index beginning in position 5, with a record length of 40. Use of this table is discussed with [Designing APIs for Multiple Record Retrieval](#).

```
//CRTALTS EXEC PGM=DK1TDRV,COND=(4,LT)
//STEPLIB DD DISP=SHR,DSN=your.prefix.tableBASE.LOAD
//TBRPT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//TBDUMP DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//MAINLIB DD DISP=SHR,DSN=your.prefix.tableBASE.MAINLIB
//TBOPT DD *
TSRSIZE=50M
/*
//CMD DD *
CA EXMPCATA,EXMPCAT,S,B,1,5,40
/*
```

When the copybook is mapped for multiple responses in z/OS Connect, it will retrieve data using EXMPCATA.

Opening a table in a VTS-TSR

The following steps can be used to open a table into a VTS-TSR.

1. Install the VTS product described in Chapter 6 of the *tableBASE Installation Guide*.
2. Upon successful installation, there must be a VTS Agent address space running.
3. Using the DK1TDRV command processor for Batch, open the table into the VTS-TSR. Details about the DK1TDRV command processor can be found in Chapter 9 in the *tableBASE Programming guide*.
4. A sample JCL along with its control card to open the table EXMPCAT into a VTS-TSR named VTS2 is provided.

```
//TBDREV1 EXEC PGM=DK1TDRV
//STEPLIB DD DISP=SHR,DSN=your.tableBASE.load
//SYSOUT DD SYSOUT=*
//MAINLIB DD DISP=SHR,DSN=your.tableBASE.library
//TBOPT DD *
TSREGION=2G
LISTOPTIONS=N
MAXNMTAB=5
//CMD DD *
SS,VTS2
OR,EXMPCAT
/*
//
```

Customize the Copybook

RESTConnector requires a specific copybook layout provided in this section.

As part of RESTConnector installation, atableBASE table called APICONFG is installed and loaded in the tableBASE Library. Following types of APIs are currently configured with the installation.

For more information about the Internal Commands please refer to the *tableBASEProgramming Guide*.

API Type	Internal Command	API Function
API001	FK	Get a single record
API002	FG	Get multiple records using a Generic Key
API003	FG	Get multiple records using a Generic Key and return hexadecimal data

Table 2-1: API Type

Customizing a Copybook to Retrieve a Single Record

tableBASE RESTConnector Interface requires a customized copybook.

The copybook should exactly match the data loaded into tableBASE table. The copybook should have standard header fields. The same can be found in your.prefix.cpycob(APISINGL) displayed in the Copybook Example.

```

01  WS-ROW-AREA.
   03  WS-ROW-HEADER.
   *    API Common request - response area
       05  IN-API-KEY                PIC X(9) .
       05  IN-TABLE-NAME             PIC X(8) .
       05  IN-VTS-NAME               PIC X(8) .
       05  IN-OFFSET                 PIC S9(9) COMP.
       05  IN-LIMIT                  PIC S9(9) COMP.
       05  OUT-TOTAL-RECS             PIC S9(8) COMP.
       05  OUT-RECS-THIS-CALL        PIC S9(8) COMP.
       05  OUT-REC-SIZE               PIC S9(8) COMP.
       05  OUT-ERROR-CODE             PIC S9(8) COMP.
       05  OUT-ERROR-SUB-CODE        PIC S9(8) COMP.
       05  OUT-RESPONSE-MESSAGE      PIC X(80) .
       05  OUT-FILLER                PIC X(67) .
   *    API Specific area
   03  WS-ROW-DATA

```

The fields related to actual data in the table's copybook will need to be indented and listed beneath the WS-ROW-DATA field.

Customizing the Copybook to Retrieve Multiple Records

In the array declaration of field WS-ROW-DATA, this API interface can retrieve a maximum of 100 records at a time.

Various header fields declared under WS-ROW-HEADER allow the API designer to customize how many records to retrieve in a single call. Define the number of records to be retrieved using IN-OFFSET and IN-LIMIT.

```

01 WS-ROW-AREA.
  03 WS-ROW-HEADER.
    * API Common request - response area
      05 IN-API-KEY PIC X(9).
      05 IN-TABLE-NAME PIC X(8).
      05 IN-VTS-NAME PIC X(8).
      05 IN-OFFSET PIC S9(9) COMP.
      05 IN-LIMIT PIC S9(9) COMP.
      05 OUT-TOTAL-RECS PIC S9(8) COMP.
      05 OUT-RECS-THIS-CALL PIC S9(8) COMP.
      05 OUT-REC-SIZE PIC S9(8) COMP.
      05 OUT-ERROR-CODE PIC S9(8) COMP.
      05 OUT-ERROR-SUB-CODE PIC S9(8) COMP.
      05 OUT-RESPONSE-MESSAGE PIC X(80).
      05 OUT-FILLER PIC X(67).
    * API Specific area
  03 WS-ROW-DATA OCCURS 100 TIMES.

```

The fields related to actual data in the table's copybook will need to be indented and listed beneath the WS-ROW-DATA field.

Customizing the Copybook to Retrieve Multiple Hexadecimal Records

Here is a sample copybook layout for generic retrieval of hexadecimal records.

```

*
01 WS-ROW-AREA.
  03 WS-ROW-HEADER.
    * API Common request - response area
      05 IN-API-KEY PIC X(9).
      05 IN-TABLE-NAME PIC X(8).
      05 IN-VTS-NAME PIC X(8).
      05 IN-OFFSET PIC S9(9) COMP.
      05 IN-LIMIT PIC S9(9) COMP.
      05 OUT-TOTAL-RECS PIC S9(8) COMP.
      05 OUT-RECS-THIS-CALL PIC S9(8) COMP.
      05 OUT-REC-SIZE PIC S9(8) COMP.
      05 OUT-ERROR-CODE PIC S9(8) COMP.
      05 OUT-ERROR-SUB-CODE PIC S9(8) COMP.
      05 OUT-RESPONSE-MESSAGE PIC X(80).
      05 OUT-FILLER PIC X(67).
    * API Specific area
  03 WS-ROW-DATA OCCURS 100 TIMES.
      05 CA-DATA PIC X(32000).

```

API Input Request Fields

Field Name	Field Description
IN-API-KEY	This field can have three possible values. API001 – For APIs that will retrieve single record. API002 – For APIs that will retrieve multiple records. API003 – For APIs that will retrieve multiple records returning data in a hexadecimal format. Note, internally tableBASE RESTConnector uses a tableBASE table called APICONFG which has all the default values for API001, API002, and API003 type APIs. This table has details like what should be the corresponding tableBASE command used by the API interface and default values used in the COMMAND-AREA used to make internal tableBASE API calls. Additional records can be added to this table APICONFG to design more complex APIs and pass more API specific COMMAND-AREA values. More details about APICONFG table can be found in <i>tableBASE Installation Guide</i>.
IN-TABLE-NAME	Name of the tableBASE table that holds the data to be exposed as API. In the DK1TEXEC example, the name of the table is EXMPCAT.
IN-VTS-NAME	This is the name of the VTS-TSR the table is loaded into. In the example given, the name of the VTS-TSR is VTS2
IN-OFFSET	This is used only with Multiple record retrieval. This is the number of records to skip from the retrieved records. Value for this field can be mapped as one of the path parameter in the HTTP GET request.

Field Name	Field Description
IN-LIMIT	This is used only with Multiple record retrieval. This is the number of records to count from the retrieved records. It is used along with the IN-OFFSET field to filter the records returned by the API.

Table 2-2: API Input Request Fields**API Output Response fields**

Field Name	Field Description
OUT-TOTAL-RECS	This field is used only with Multiple record retrieval. This contains the total number of records retrieved by the HTTP GET request. The value for this field can be mapped to an HTTP Header field as a response to the HTTP GET request
OUT-RECS-THIS-CALL	This field is used only with Multiple record retrieval. This contains the number of records returned by the current API call, not the total records retrieved by the HTTP GET request. The total number of records retrieved by the API can be filtered using the IN-OFFSET and IN-LIMIT fields. This field is used as a COUNTED BY field to limit the array size of the API response.
OUT-REC-SIZE	This contains the size of the input data layout received by the API interface.
OUT-ERROR-CODE	This contains the EIBRESP response code if any of the CICS command API calls failed or if a TB-ERROR-CODE if tableBASE API call failed. This field can be used to diagnose the error condition. All tableBASE error codes and their description can be found in <i>tableBASE Programming Guide</i> Appendix E, tableBASE Error Codes.
OUT-ERROR-SUB-CODE	This contains the EIBRESP2 response code if any of the CICS command API calls failed or if a TB-ERROR-SUB-CODE if tableBASE API call failed. This field can be used to diagnose the error condition. All tableBASE error sub codes and their description can be found in <i>tableBASE Programming Guide</i> Appendix E, tableBASE Error Codes.

Field Name	Field Description
OUT-RESPONSE-MESSAGE	This field can be used to diagnose the error condition.
OUT-FILLER	This field is available for future enhancements.

Table 2-3: API Output Response Fields

Chapter

3

Designing APIs for Single Record Retrieval

This Chapter explains how to design and customize APIs using IBM Explorer for z/OS to create services and REST APIs for accessing z/OS resources.

The steps include:

1. [Create a Service Project.](#)
2. [Deploy the Service Project.](#)
3. [Create and configure an API Project.](#)
4. [Map GET Method for the request to the API.](#)
5. [Map GET Method for the response from the API.](#)
6. [Deploy the API Project.](#)

Note: The names of all tableBASEtables and VTSeS are for test purposes. Please use your organization's naming schema.

Prerequisites

z/OS Connect V3.0 or later is installed.

IPIC connection to CICS has been configured.

Data is loaded into a tableBASE table using a VTS-TSR.

Customization of corresponding copybooks is completed.

Set Perspective

To view the z/OS Connect EE Service Project, set the perspective to the OpenAPI 2 perspective

1. Start IBM Explorer for z/OS and open the z/OS Connect Enterprise Edition (OpenAPI 2) perspective. Select **Window > Select Perspective > Select Open Perspective > Select Other**. The Open Perspective window will display.

2. Select **z/OS Connect Enterprise Edition** and the Open button.

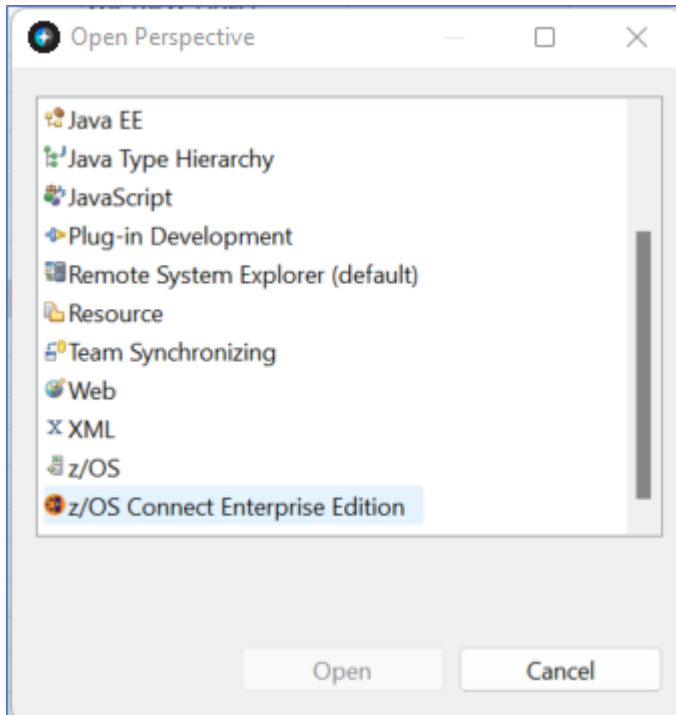


Figure 3-1: Open Perspective Window

Create a New z/OS Connect EE Service Project

A service interface defines how the underlying program or COBOL copybook can be accessed, which fields can be exposed, and how they are exposed.

z/OS Connect supports the following tasks with services:

- Deployment
- Starting and stopping
- Deleting
- Retrieving details and statistics

For service archives, configure the server for automatic service deployment.

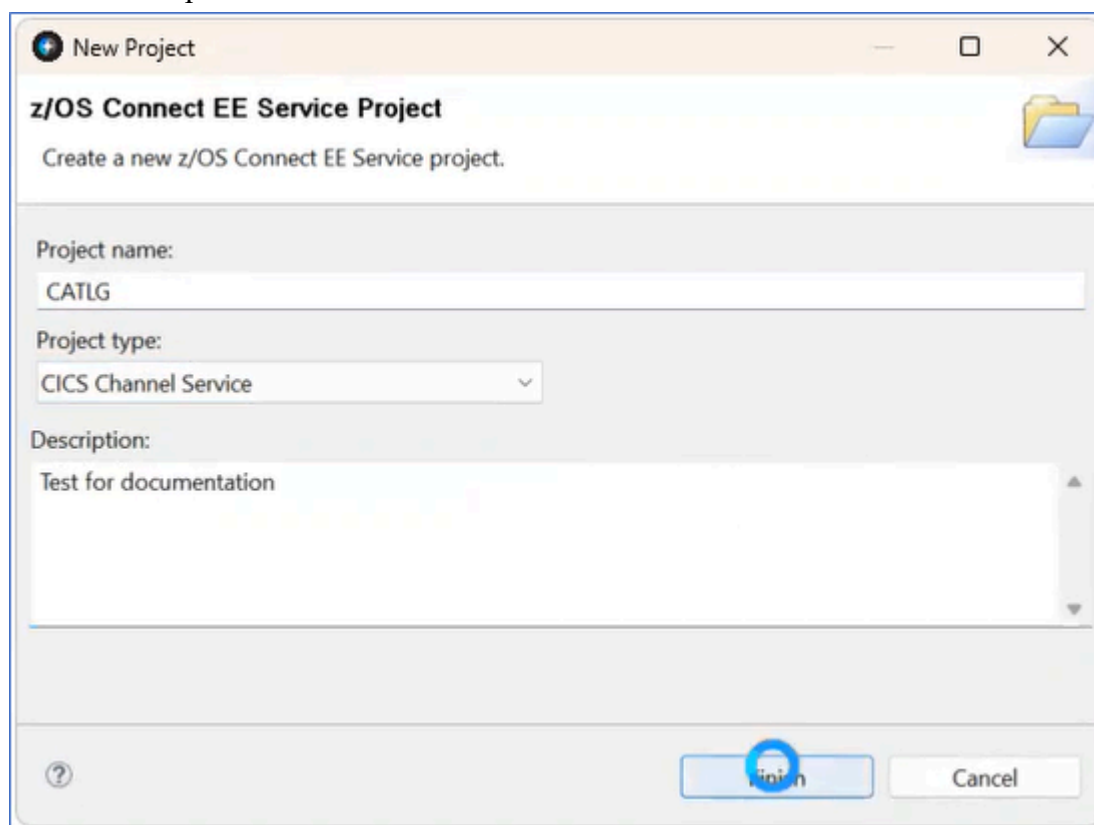
To implement RESTConnector, define a Service Project. Additional information using IBM Explorer for z/OS can be found in z/OS Connect (OpenAPI 2) 3.0 documentation.

Create a New Service Project

1. Select **File > New > z/OS Connect EE Service Project**. The [New Project Window](#) will display.
2. Enter the following details:
 - a. Project Name is **CATLG**

Note: CATLG is the name provided for this sample. Please use your organization's naming schema.

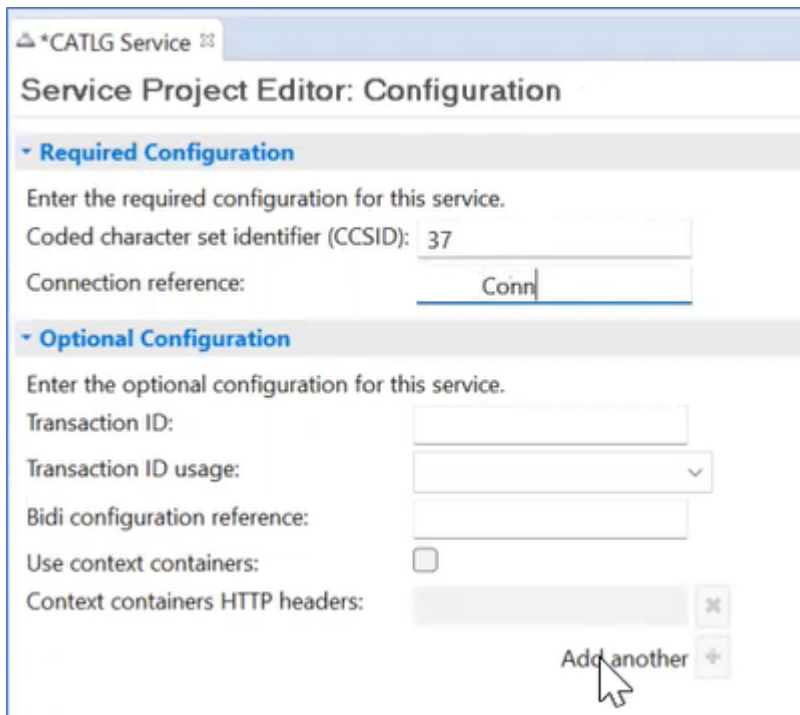
- b. Project Type is **CICS Channel Service**.
- c. Enter a description.



The screenshot shows a 'New Project' dialog box with the title 'z/OS Connect EE Service Project'. Below the title is the instruction 'Create a new z/OS Connect EE Service project.' and a folder icon. The dialog contains three input fields: 'Project name:' with the text 'CATLG', 'Project type:' with a dropdown menu showing 'CICS Channel Service', and 'Description:' with the text 'Test for documentation'. At the bottom, there is a question mark icon, a 'Finish' button (highlighted with a blue circle), and a 'Cancel' button.

Figure 3-2: New Project Window for Service Project

3. Select the **Configuration** tab for the CATLG Service. In the [Service Project Editor](#), enter the CICS Connection in the Connection reference field.

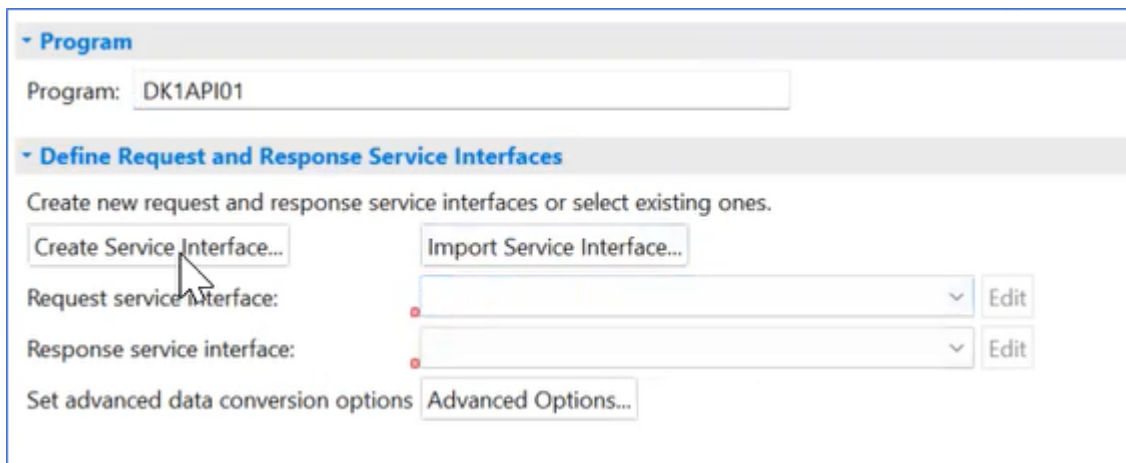


The screenshot shows the 'Service Project Editor: Configuration' window for the '*CATLG Service'. It is divided into two main sections: 'Required Configuration' and 'Optional Configuration'. In the 'Required Configuration' section, the 'Coded character set identifier (CCSID)' is set to '37' and the 'Connection reference' is 'Conn'. The 'Optional Configuration' section includes fields for 'Transaction ID', 'Transaction ID usage' (a dropdown menu), 'Bidi configuration reference', 'Use context containers' (a checkbox), and 'Context containers HTTP headers' (a text area with a clear button). An 'Add another' button with a plus icon is located at the bottom right of the optional configuration section.

Figure 3-3: Service Project Editor

Create New Service Interface

1. The [Service Project Editor: Definition tab](#) for the CATLG Service will display. In the Program field, enter Program name DK1API01.
2. Select the [Create Service Interface](#) button in the Define Request and Response Service Interfaces section. The [Create Service Interface window](#) will display.



The screenshot shows the 'Service Project Editor: Definition' window for the '*CATLG Service'. The 'Program' field is set to 'DK1API01'. Below this is the 'Define Request and Response Service Interfaces' section, which contains the instruction 'Create new request and response service interfaces or select existing ones.' There are two buttons: 'Create Service Interface...' and 'Import Service Interface...'. Below these are two rows for 'Request service interface' and 'Response service interface', each with a dropdown menu and an 'Edit' button. At the bottom, there is a 'Set advanced data conversion options' label and an 'Advanced Options...' button.

Figure 3-4: Create Service Interface Button

3. In the Service Interface Name enter **CATLG** Select the OK button. The [Create Service Interface window](#) will close. A new tab for CATLG will display in the [Service Interface Editor](#).

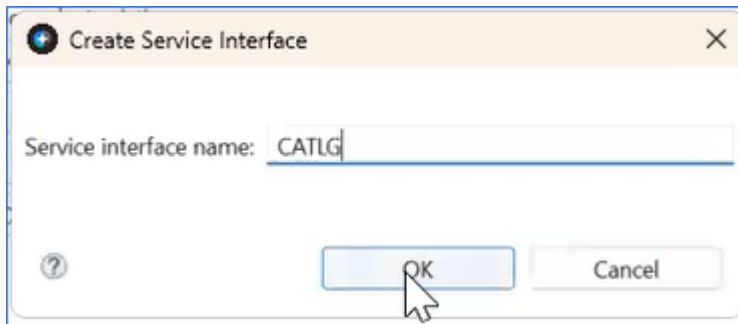


Figure 3-5: Create Service Interface window

4. Right-click to select the **Container1** > **Edit Container**. The [Edit Container window](#) will display.

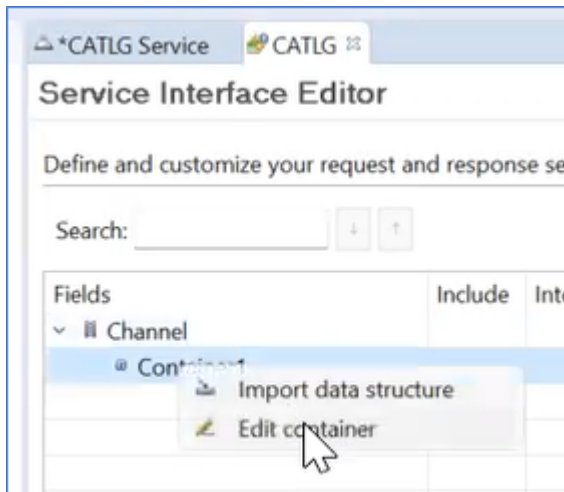


Figure 3-6: CATLG Edit Container

5. Enter **DK1APIC1-Con** in the Container name field. Select the OK button. The [Edit Container window](#) will close.

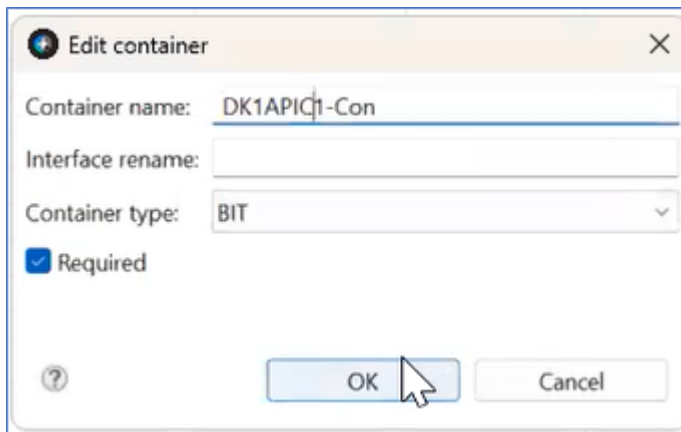


Figure 3-7: Edit Container Window

6. In the [Service Interface Editor](#), right-click to select **DK1APIC1-Con > Import Data Structure**. The [Import Window](#) will display.

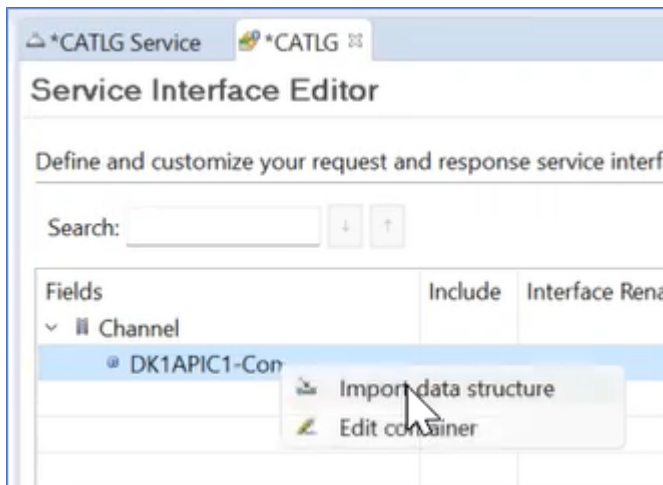


Figure 3-8: Import data structure

7. In the [Import Window](#)
 - a. The Import from field is automatically populated with **z/OS system**.
 - b. The File type field is **COBOL data structures only**.
 - c. Select the **Source PDS**.

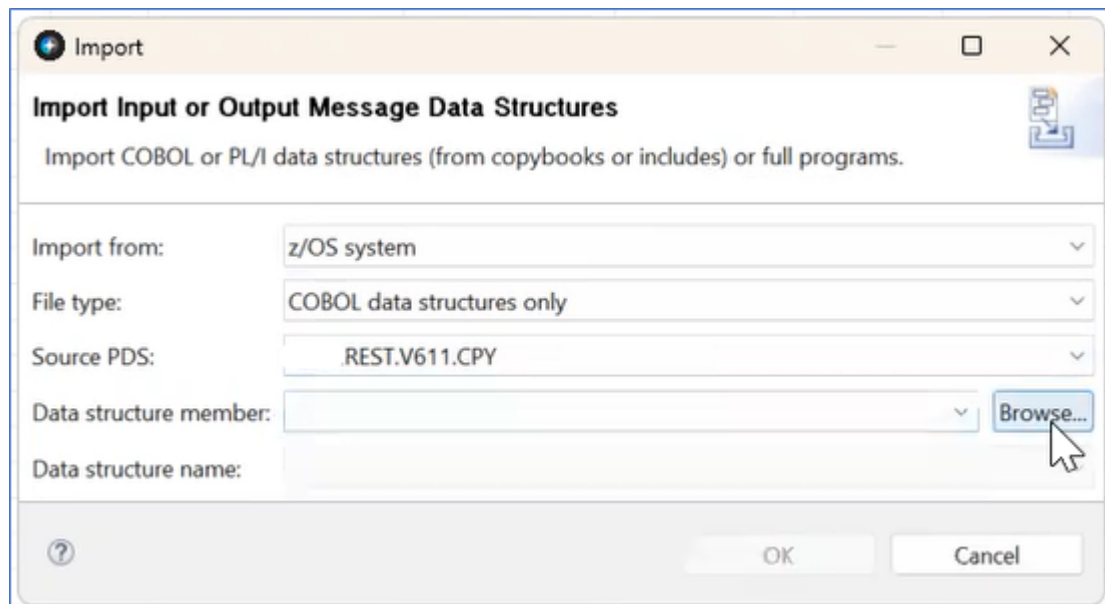


Figure 3-9: Import Window

- d. Select the **Browse** button. The [Import Application Data Structure Source window](#) displays.
 - 1) Enter the Data Set Name or pattern.
 - 2) Expand the Data Set Name to display the copybooks.

- 3) Select the **CATLG** copybook.

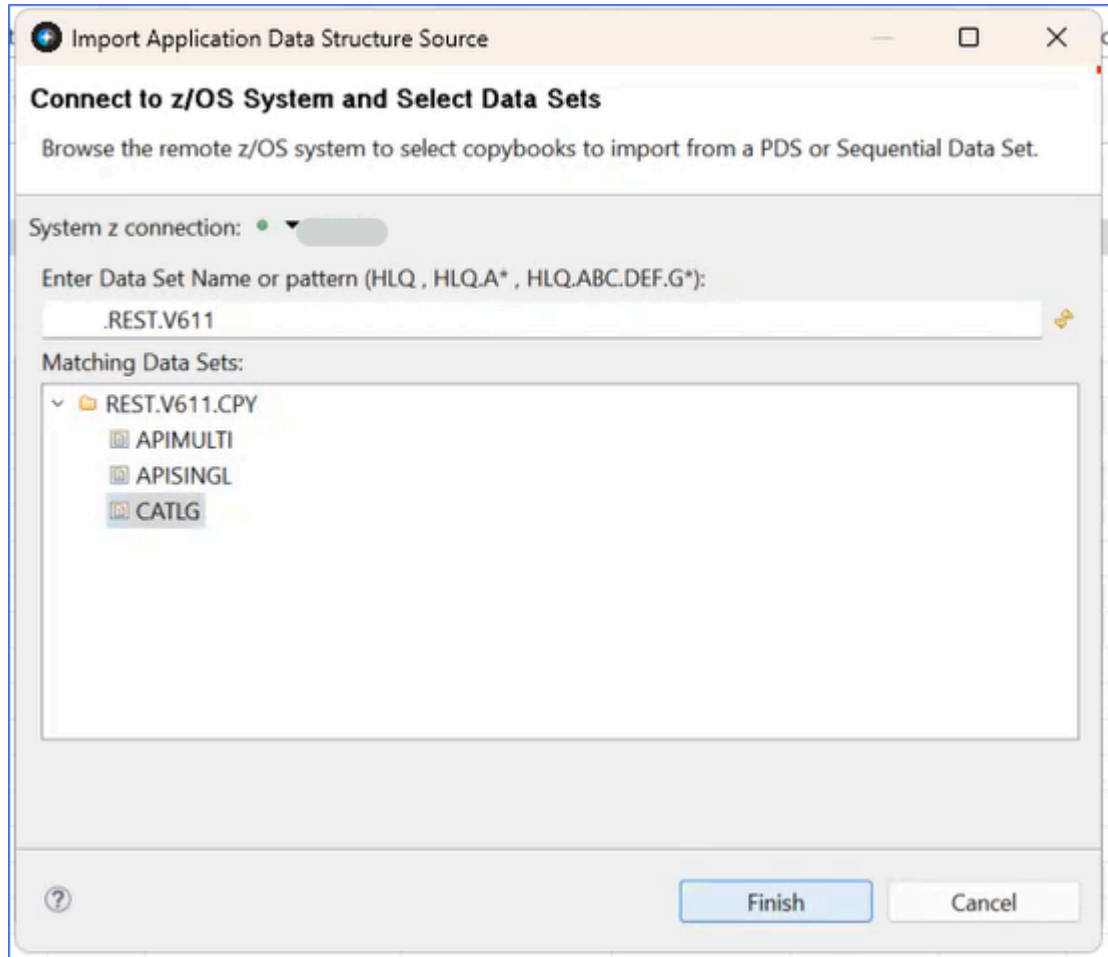


Figure 3-10: Import Application Data Structure Source window

- 4) Select the **Finish** button. The [Import window](#) will be populated with the Data Structure member and the Data Structure name.

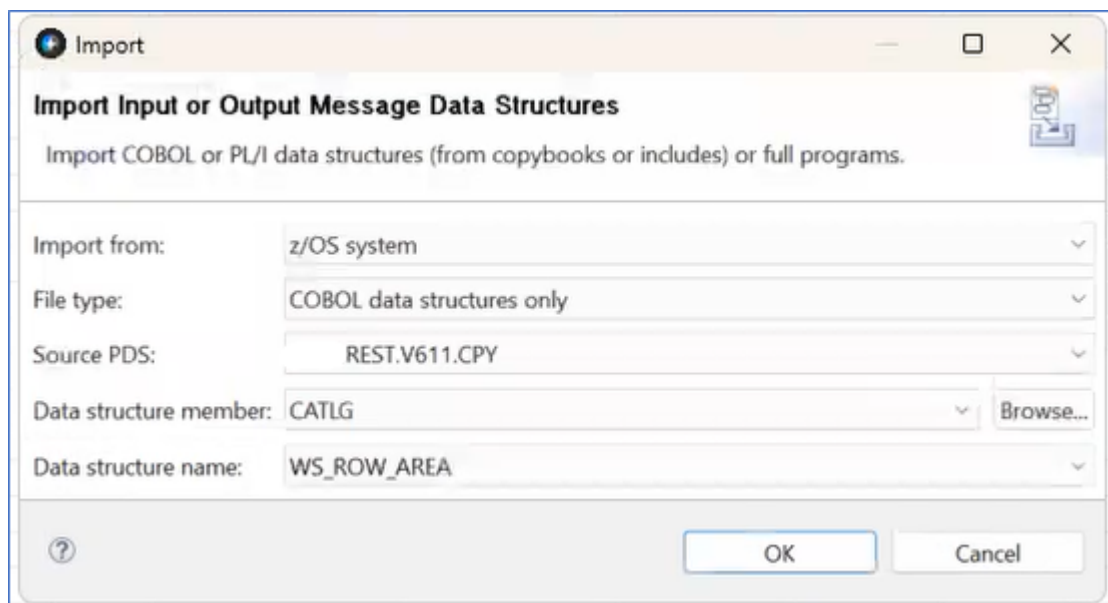
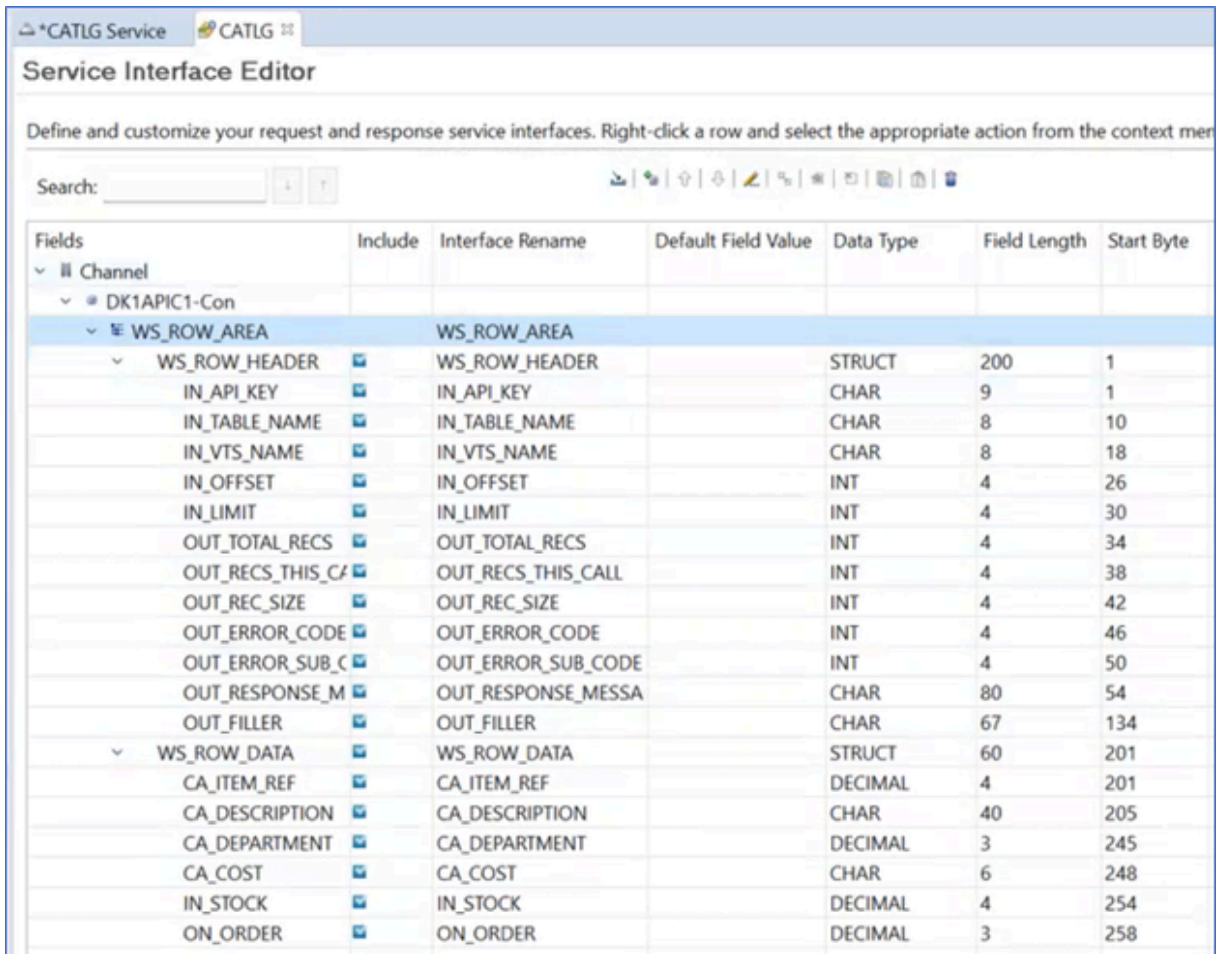


Figure 3-11: Import Window populated

- 5) Select the OK button. The [Import window](#) will close.
8. On the [Service Interface Editor](#), expand DK1APIC1-Con to display the whole copybook. The WS_ROW_HEADER shows the RESTConnector specific information for the copybook. The WS_ROW_DATA is the related table data.



Fields	Include	Interface Rename	Default Field Value	Data Type	Field Length	Start Byte
Channel						
DK1APIC1-Con						
WS_ROW_AREA		WS_ROW_AREA				
WS_ROW_HEADER	☒	WS_ROW_HEADER		STRUCT	200	1
IN_API_KEY	☒	IN_API_KEY		CHAR	9	1
IN_TABLE_NAME	☒	IN_TABLE_NAME		CHAR	8	10
IN_VTS_NAME	☒	IN_VTS_NAME		CHAR	8	18
IN_OFFSET	☒	IN_OFFSET		INT	4	26
IN_LIMIT	☒	IN_LIMIT		INT	4	30
OUT_TOTAL_RECS	☒	OUT_TOTAL_RECS		INT	4	34
OUT_RECS_THIS_C/	☒	OUT_RECS_THIS_CALL		INT	4	38
OUT_REC_SIZE	☒	OUT_REC_SIZE		INT	4	42
OUT_ERROR_CODE	☒	OUT_ERROR_CODE		INT	4	46
OUT_ERROR_SUB_C	☒	OUT_ERROR_SUB_CODE		INT	4	50
OUT_RESPONSE_M	☒	OUT_RESPONSE_MESSA		CHAR	80	54
OUT_FILLER	☒	OUT_FILLER		CHAR	67	134
WS_ROW_DATA	☒	WS_ROW_DATA		STRUCT	60	201
CA_ITEM_REF	☒	CA_ITEM_REF		DECIMAL	4	201
CA_DESCRIPTION	☒	CA_DESCRIPTION		CHAR	40	205
CA_DEPARTMENT	☒	CA_DEPARTMENT		DECIMAL	3	245
CA_COST	☒	CA_COST		CHAR	6	248
IN_STOCK	☒	IN_STOCK		DECIMAL	4	254
ON_ORDER	☒	ON_ORDER		DECIMAL	3	258

Figure 3-12: DK1APIC1-Con Expanded

Deploy the Service to the Server

1. Using the right mouse button select CATLG > z/OS Connect EE > Deploy Service to z/OS Connect EE Server. The [Deploy Service Window](#) will display.

2. Select the OK button. The [Deploy Service Window](#) will close. The [Deploy Service to z/OS Connect EE Server Result window](#) will display.

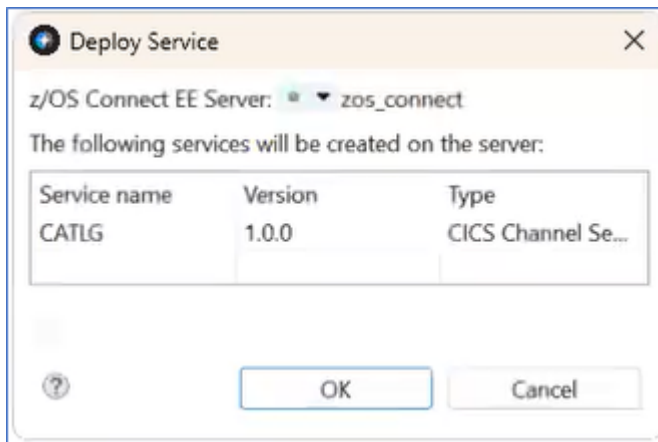


Figure 3-13: Deploy Service

3. Select the OK button. The [Deploy Service to z/OS Connect EE Server Result window](#) will close.

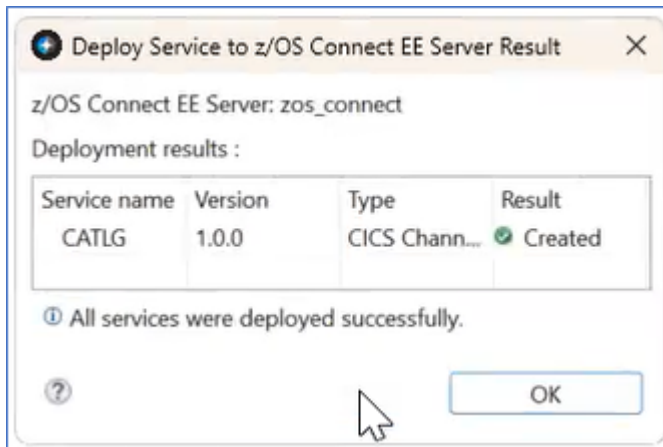


Figure 3-14: Deploy Service to z/OS Connect EE Server Result

4. In the [z/OS Connect EE Servers](#) tab, the Service CATLG displays under the Services folder.

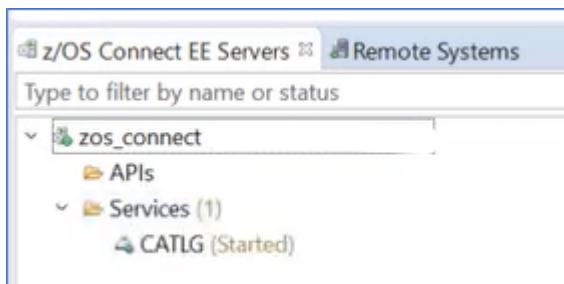


Figure 3-15: CATLG z/OS Connect EE Servers

Create a New API Project

Using the Service Archive (.sar) file for the service: create, export and deploy the API Project.

Instructions to create a new z/OS API Project can be found on IBM's website at [Invoke CICS Catalog Manager Services](#)

1. Select **File > New > z/OS Connect EE API Project**. The [New Project window](#) will display.
2. In the [New Project window](#) enter the following information:
 - a. The Project name is **CATLGAPI**.
 - b. the API Name will populate with **catlgapi**. It is an editable field.
 - c. The Base Path will populate with **/catlgapi**. It is an editable field.
 - d. Enter an optional description.
 - e. Select the **Finish** button. The [New Project window](#) will close and the [API Editor](#) will display.

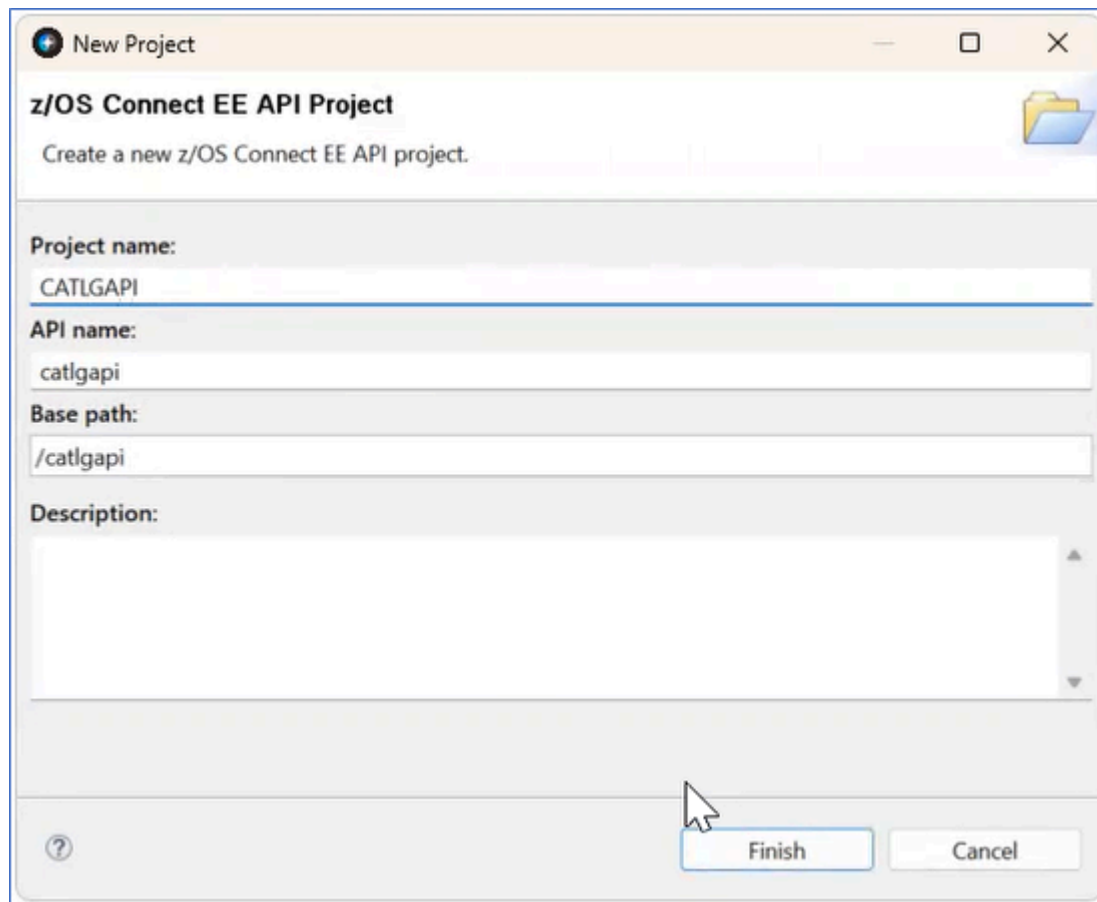


Figure 3-16: New API Project

3. In the [API Editor](#), remove the **POST**, **PUT**, and **DELETE** methods by selecting the box with a red X to the right of the Method. **GET** should be the only remaining Method.

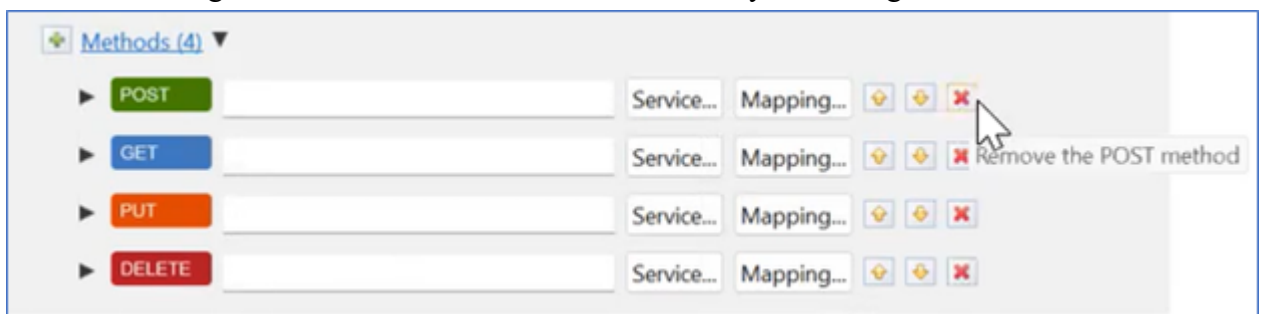


Figure 3-17: API Editor with Four Methods before removal

4. In the [Path field](#), change the path name to `/items/{itemsID}`

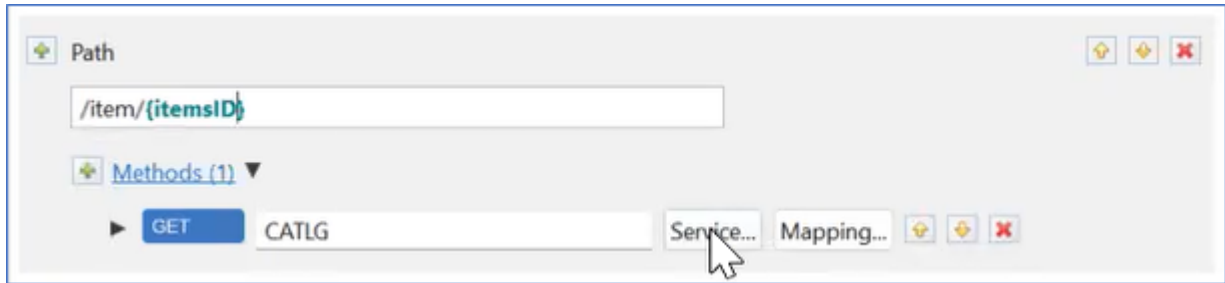


Figure 3-18: Path equals itemsID

5. In the Project Explorer panel, right-click to select the **CATLG > z/OS Connect EE > Export z/OS Connect EE Service Archive**. The [Export Service Package window](#) displays.

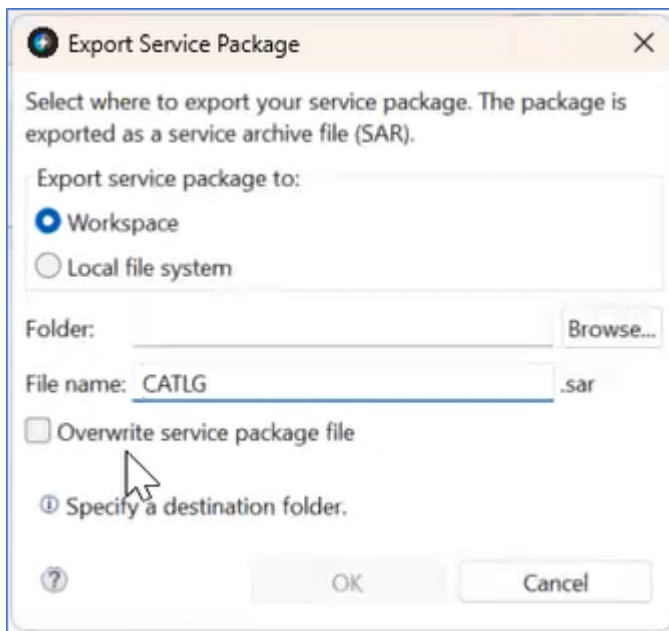


Figure 3-19: Export Service Package Window

6. Select the **Browse** button. The [Select Workspace Folder](#) window will display.

- a. Select CATLGAPI.

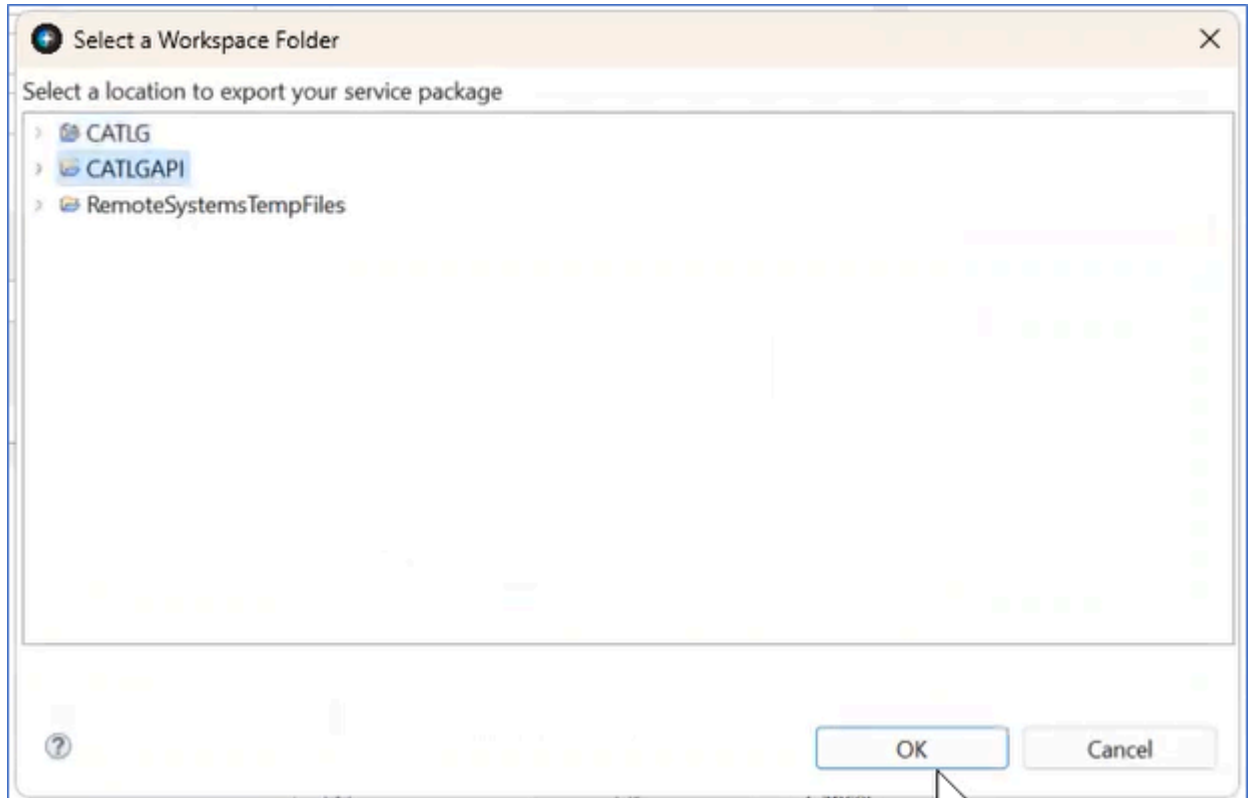


Figure 3-20: Select a Workspace Folder

- b. Select the OK Button. The [Select Workspace Window](#) will close. The [Export Service Package window](#) displays the Folder is mapped to /CATLGAPI.

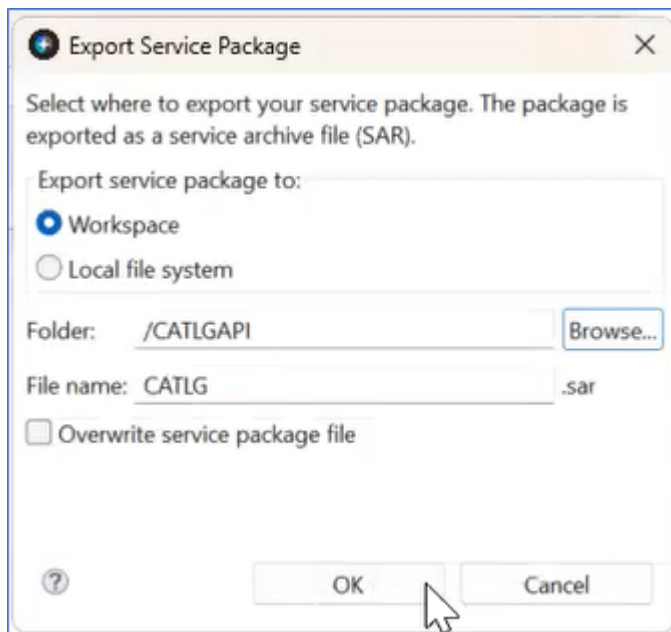


Figure 3-21: CATLGAPI Export Service Package window

- c. Select the OK button. The [Export Service Package window](#) closes. The CATLG.sar file is ready to be exported.

Import the Service Archive

More information about exporting the service archive can be found on IBM's website [Service Exporting Project](#).

1. Select the **Service** button to the right of the [GET Method](#). The [Select z/OS Connect EE Service window](#) will display.

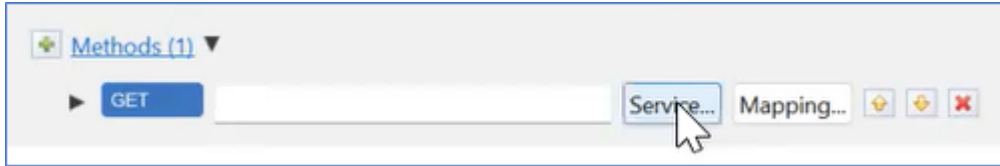


Figure 3-22: GET Method

2. In the [Select z/OS Connect EE Service window](#) select the **Workspace** button. The [Import z/OS Connect EE Services window](#) will display.

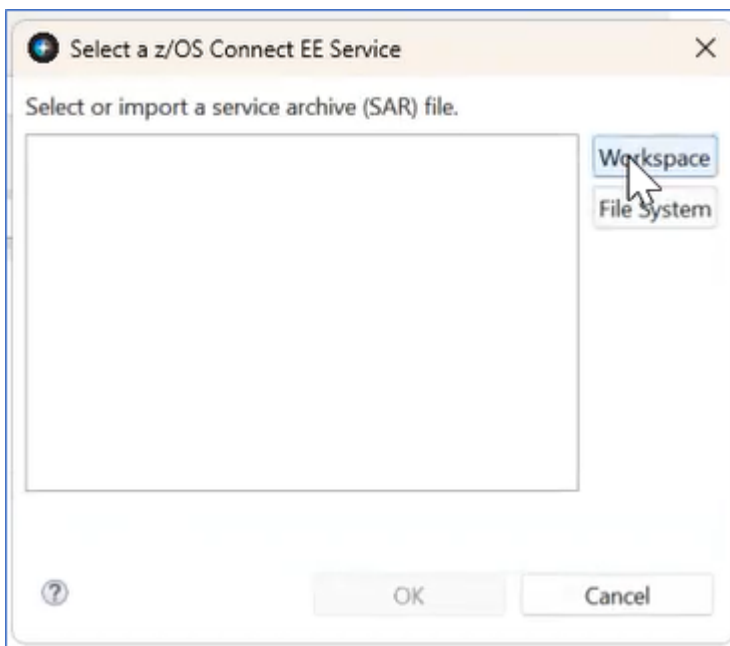


Figure 3-23: Select zOS Connect EE Service window

3. In the [Import z/OS Connect EE Services window](#):
 - a. Expand the **CATLGAPI** folder.
 - b. Select the **CATLG.sar** file.

- c. Select the OK button. The [Import z/OS Connect EE Services window](#) will close. The [Import Services window](#) will display.

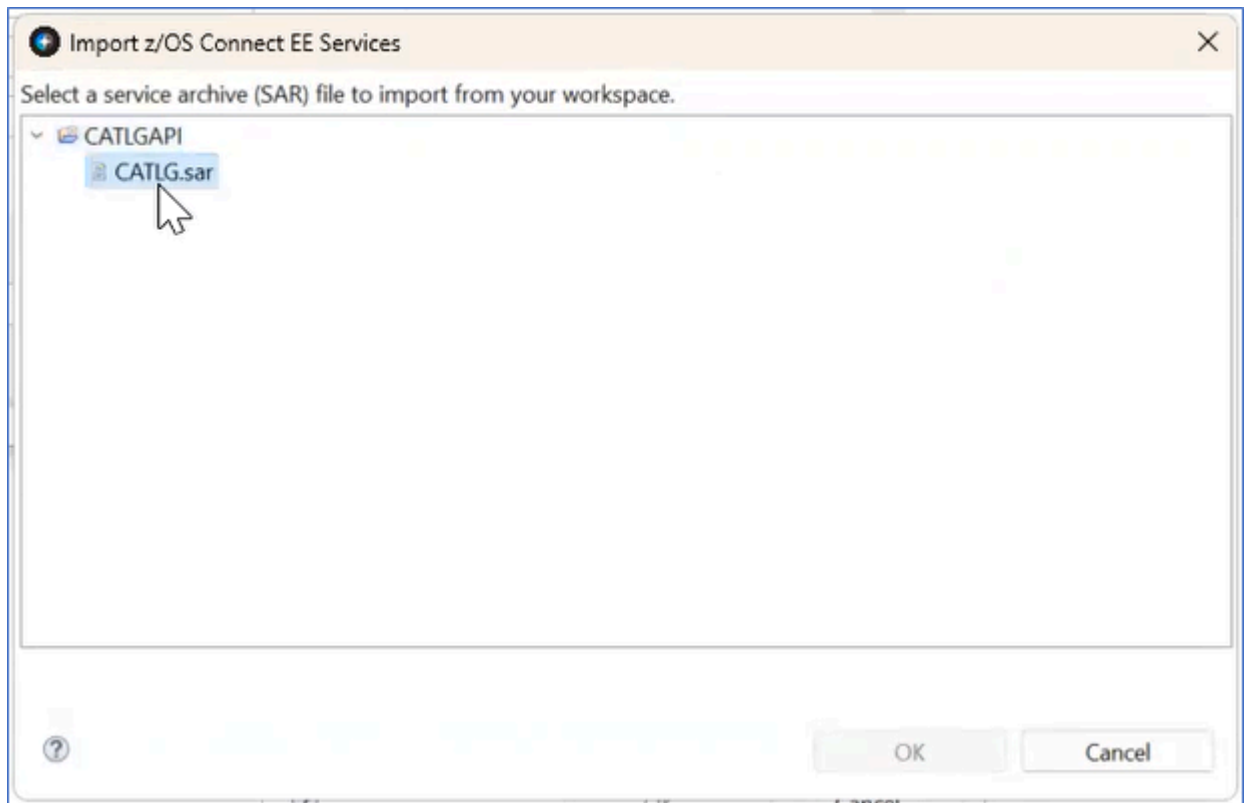


Figure 3-24: Import z/OS Connect EE Services window

4. In the [Import Services window](#) the Service name and description display. Select the OK button. The [Import Services window](#) will close. The [Select z/OS Connect EE Service window](#) displays the CATLG.sar file is selected.

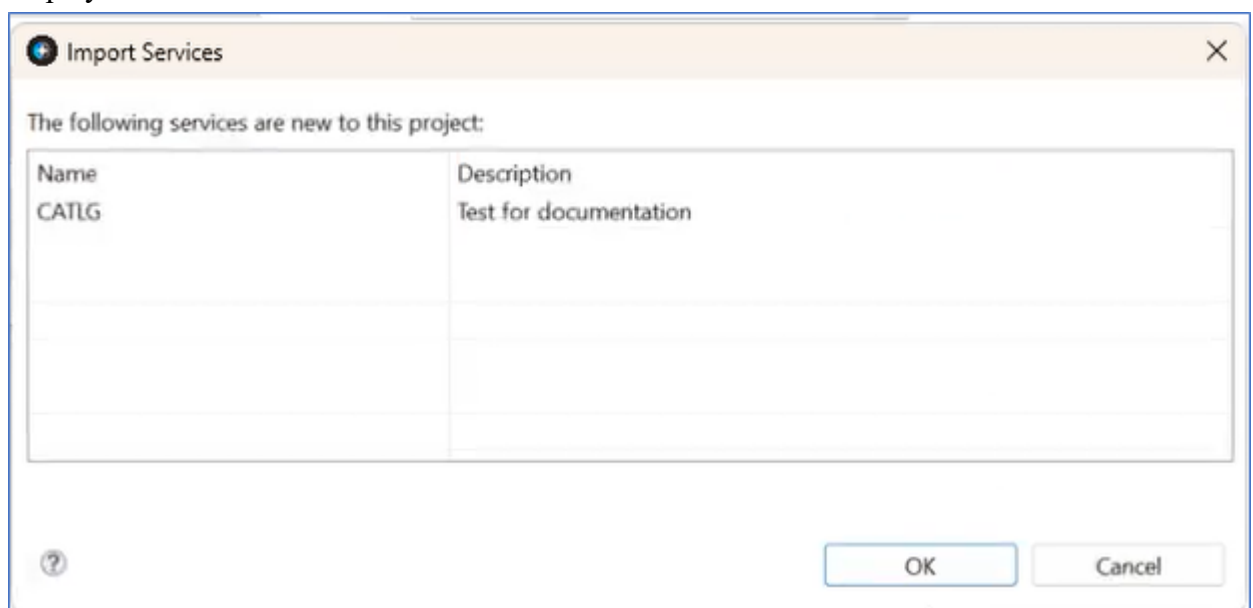


Figure 3-25: Import Services window

5. In the [z/OS Connect EE Service window](#) select the OK button. The window will close. The GET service, CATLG, is mapped to the [GET Method](#).

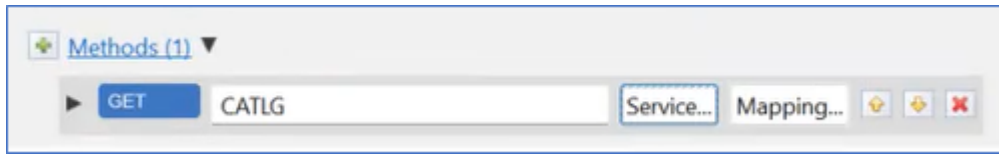


Figure 3-26: GET Service mapped

Mapping the Request

The following steps create an operation to GET an item from the EXMPCAT table. For additional information refer to [IBM z/OS Connect](#) topic regarding **Create an Operation to Inquire about an item**.

1. Select the **Mapping button** > [Open Both Mappings](#) next to the GET Method Service button.

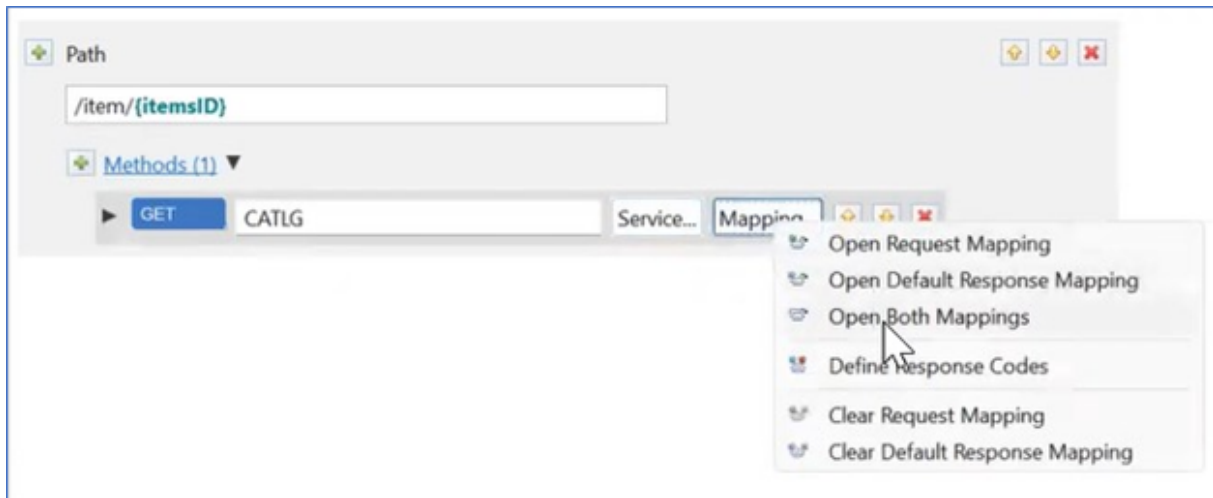


Figure 3-27: Open Both Mappings

2. A [request tab and response tab](#) will display next to the catlgapi API Editor tab. Select the **request** tab.

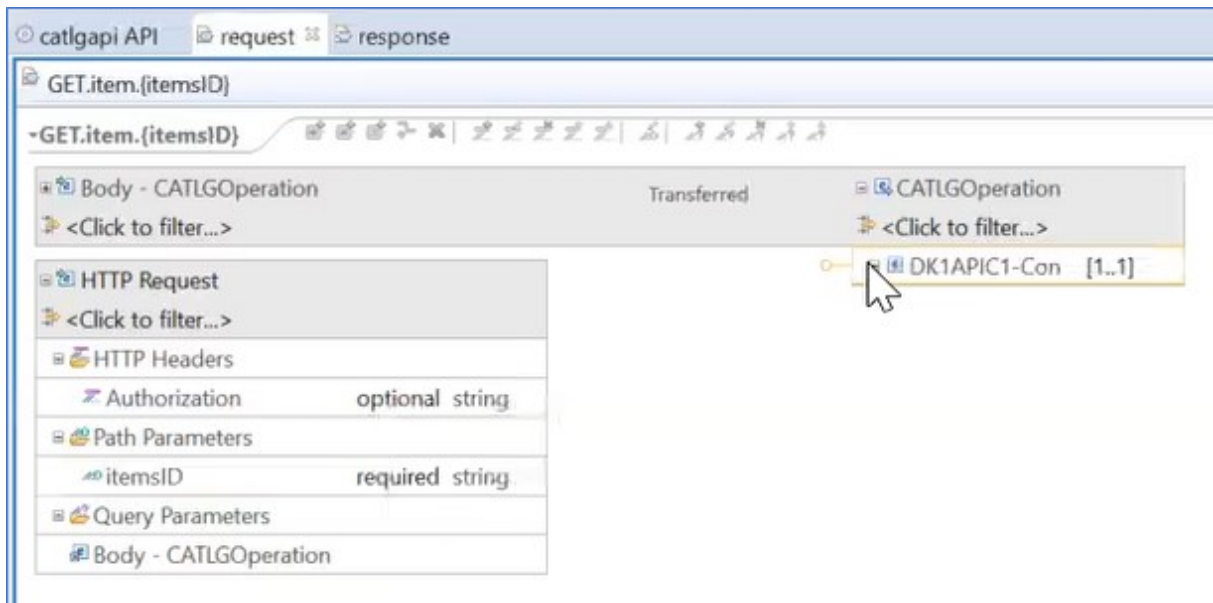


Figure 3-28: Request and Response Tabs

3. Expand the DK1APIC1-Con to display the whole copybook.
4. This step will assign the API Key. Right-click to select the **IN_API_KEY** field in the copybook.

- a. Select [Add Assign transform](#).

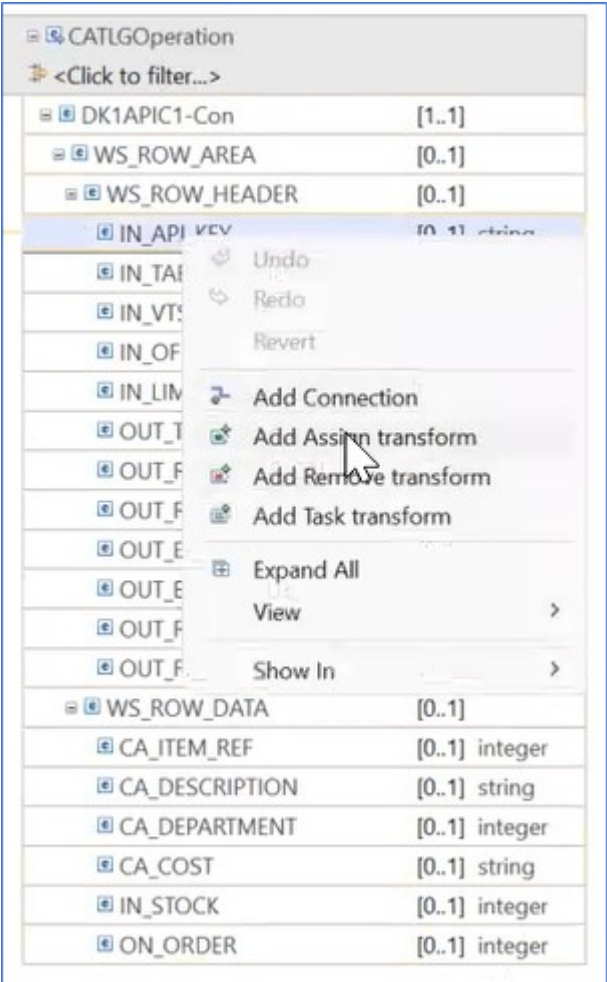


Figure 3-29: Add Assign Transform

- b. In the [Properties Panel](#) enter the value **API001**. A box displays next to the field **IN_API_KEY** marking it as **Assigned**.

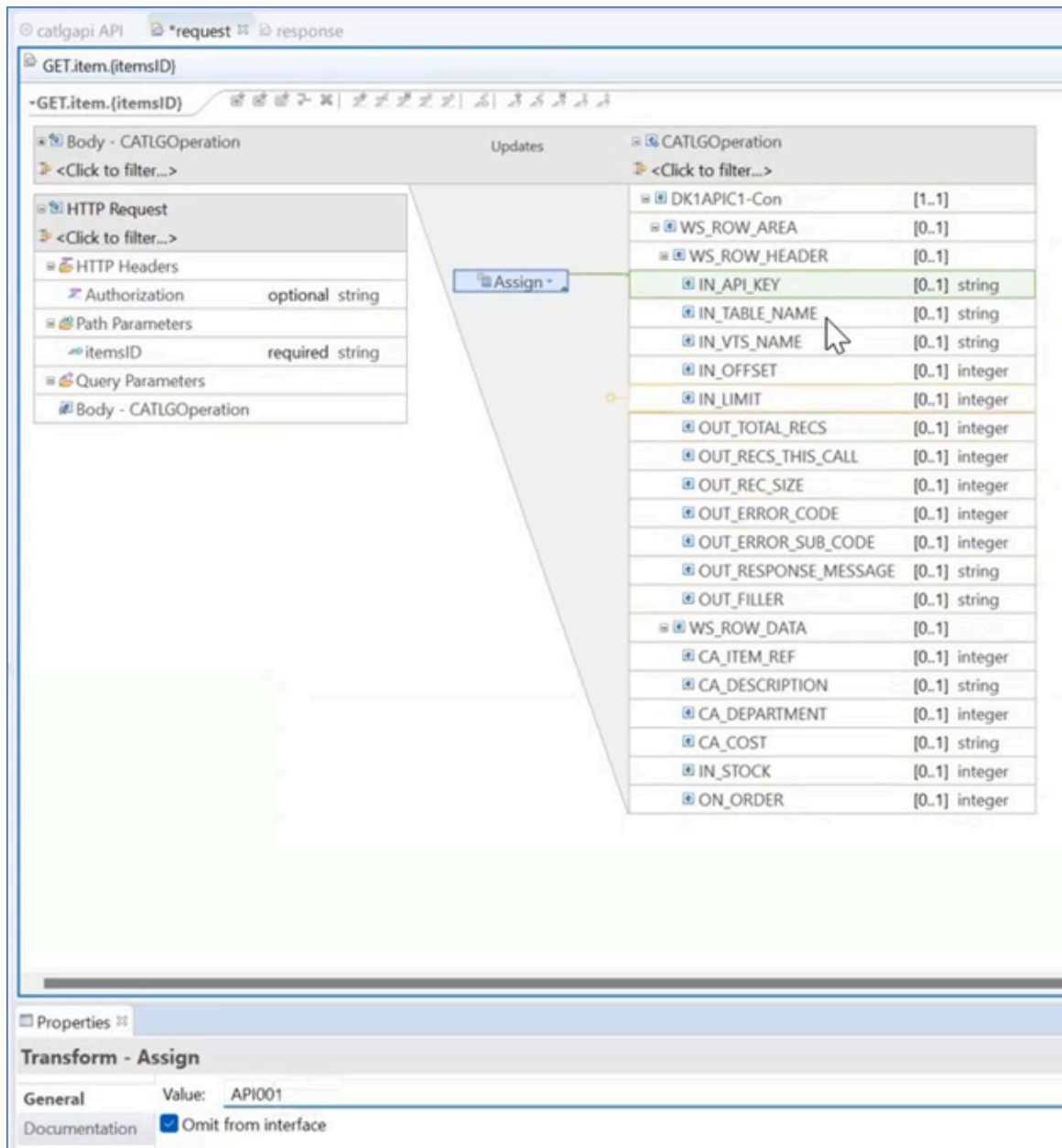


Figure 3-30: Assign IN_API_KEY

5. This step will assign the table name. Right-click to select the **IN_TABLE_NAME** field in the copybook.
- a. Select [Add Assign transform](#).

- b. In the [Properties Panel](#) enter the table name **EXMPCAT**. A box displays next to the field marking it as **Assigned**.

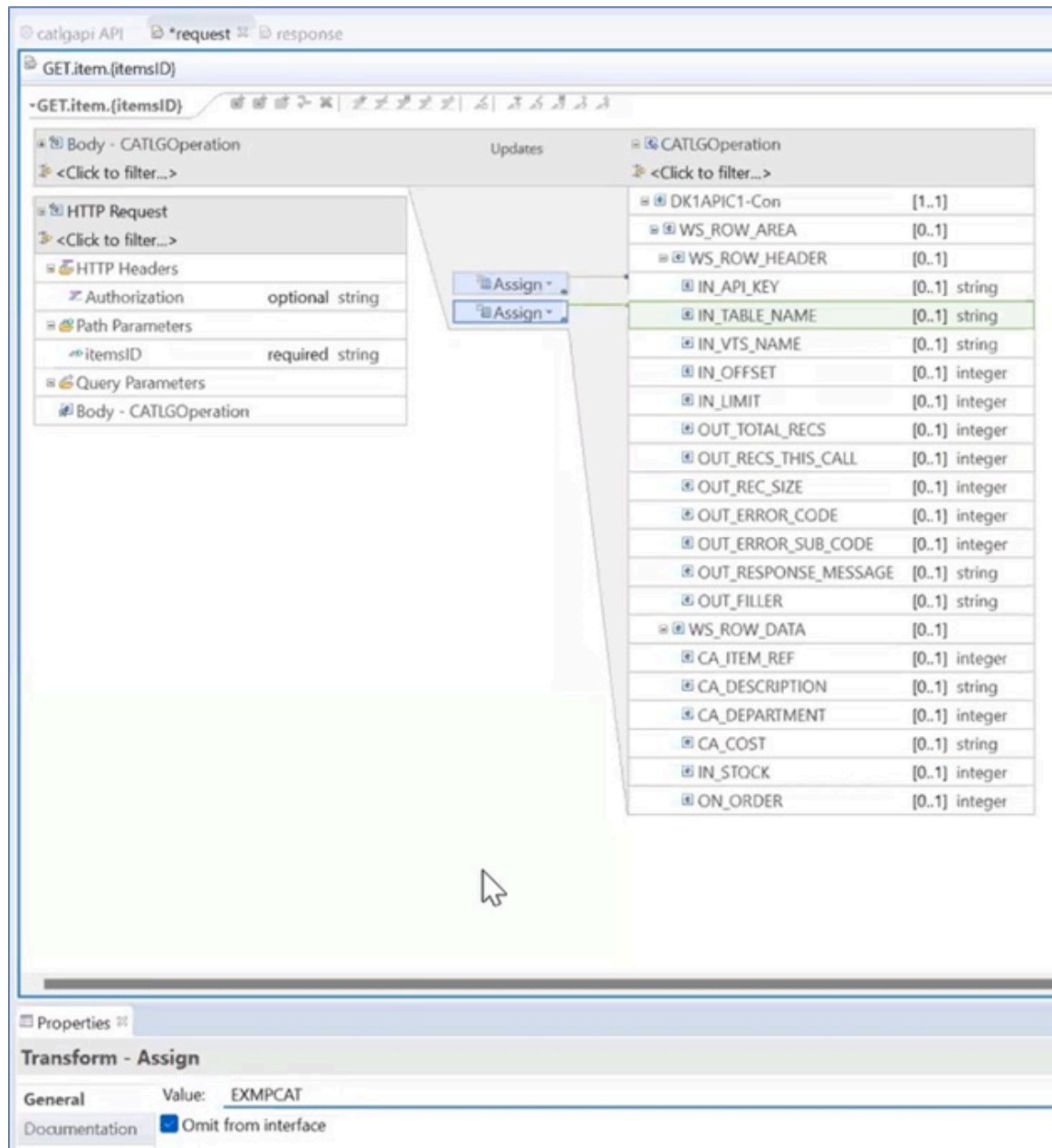


Figure 3-31: Assign Table Name

6. This step will assign the VTS. Right-click to select the **IN_VTS_NAME** field in the copybook.

Note: If using a local TSR, this field can be ignored.

- a. Select [Add Assign transform](#).

- b. In the [Properties panel](#) enter the name of the VTS-TSR. A box displays next to the field marking it as **Assigned**.

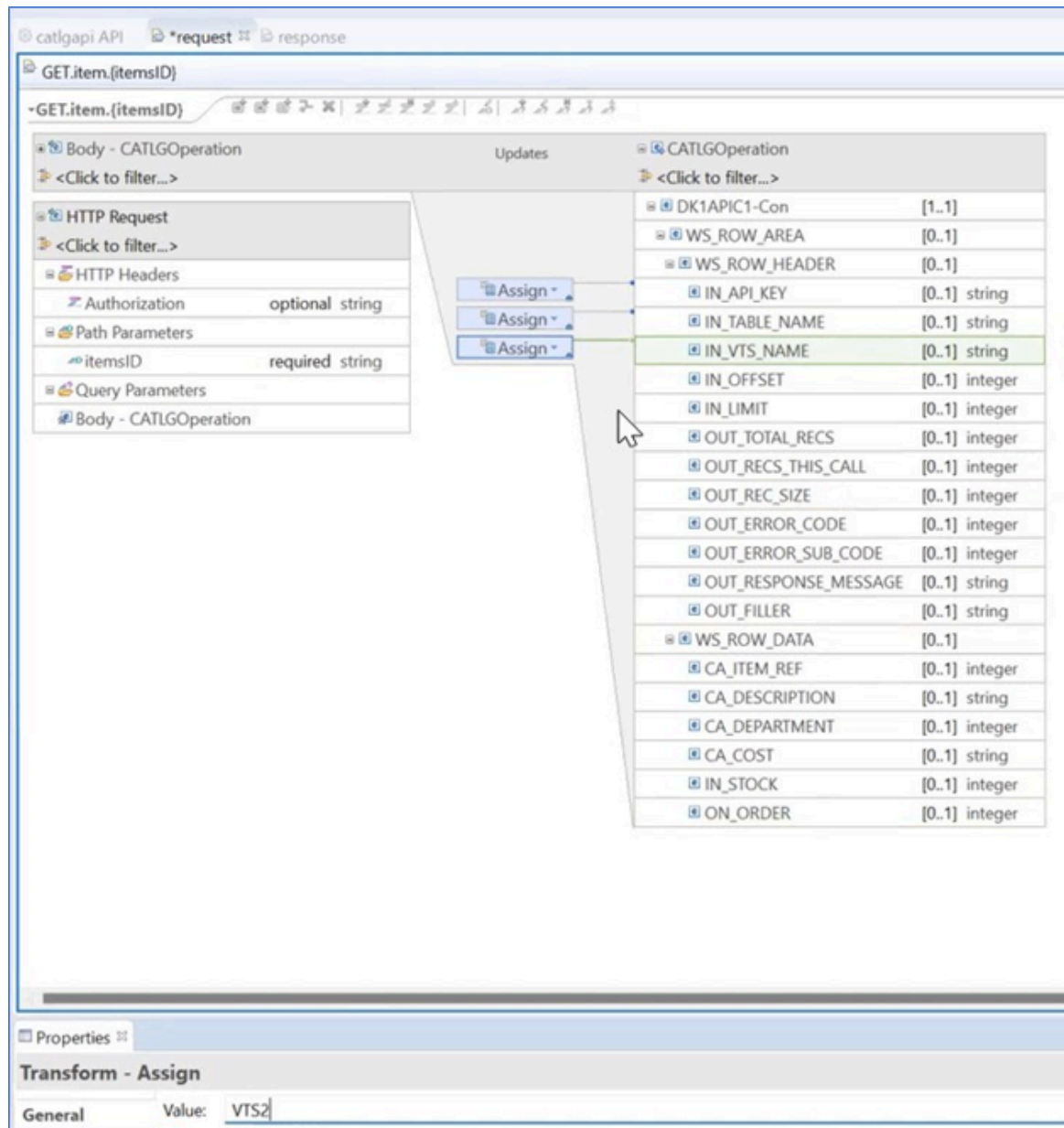


Figure 3-32: Assign VTS Name

7. Select the following fields in the **WS_ROW_HEADER** section:

- IN_OFFSET
- IN_LIMIT
- OUT_TOTAL_RECS
- OUT_RECS_THIS_CALL
- OUT_REC_SIZE
- OUT_ERROR_CODE
- OUT_ERROR_SUB_CODE
- OUT_RESPONSE_MESSAGE

- OUT_FILLER
8. Right click to select **Add Remove Transform**. A box displays next to the field marking it as **Removed**.

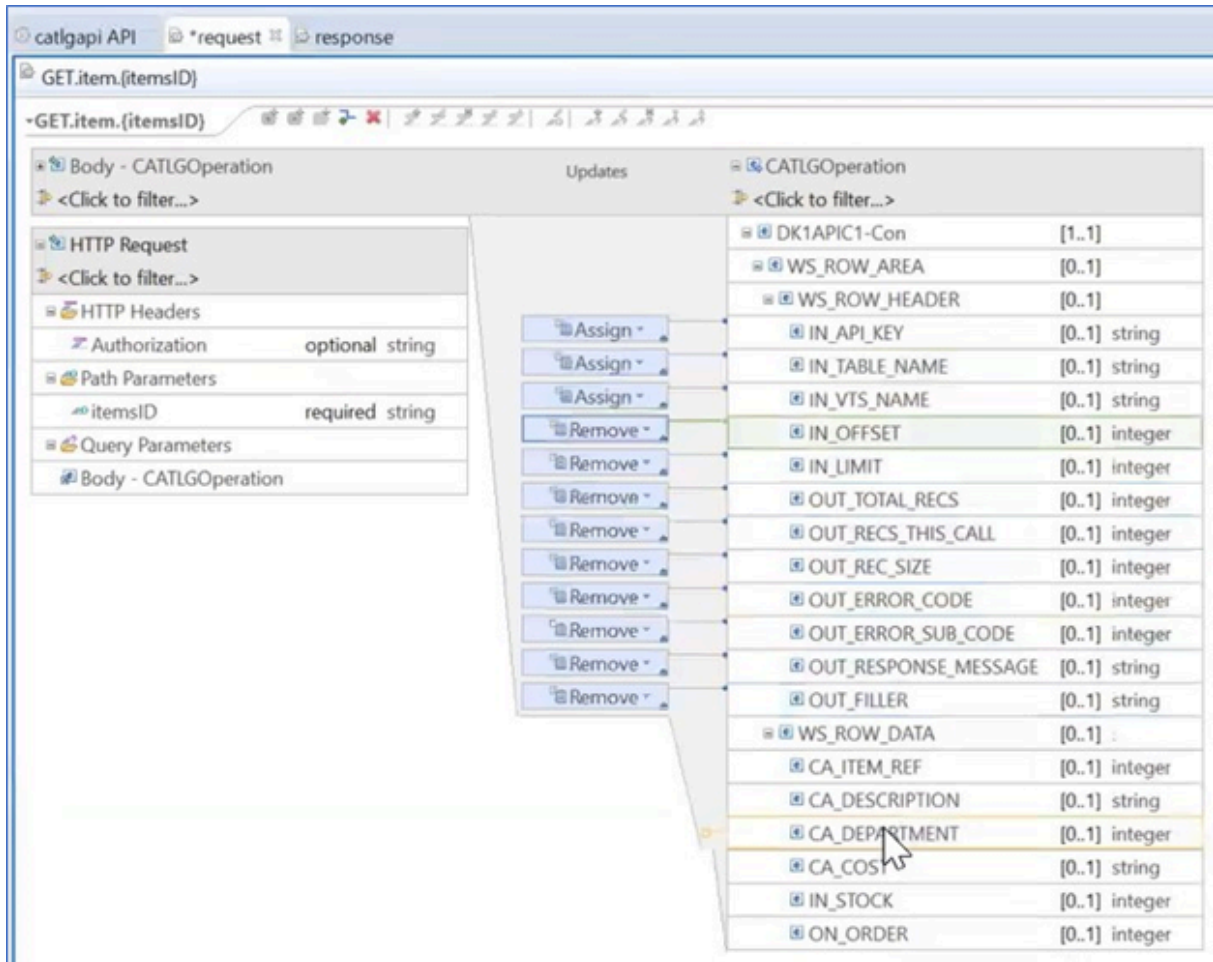


Figure 3-33: Add Remove Transform

9. Select the following fields associated with the request in the **WS_ROW_DATA** section:

- CA_DESCRIPTION
- CA_DEPARTMENT
- CA_COST
- CA_STOCK
- IN_STOCK
- ON_ORDER

10. Select **Add Remove Transform**. A box displays next to each field marking it as **Removed**.

11. In the [HTTP Request section](#), select the **itemsID** listed as a Path Parameter and drag it to the **CA_ITEM_REF** field in the **WS_ROW_DATA** section. A box displays **Move** between the Path Parameter and the CA_ITEM_REF Request field.

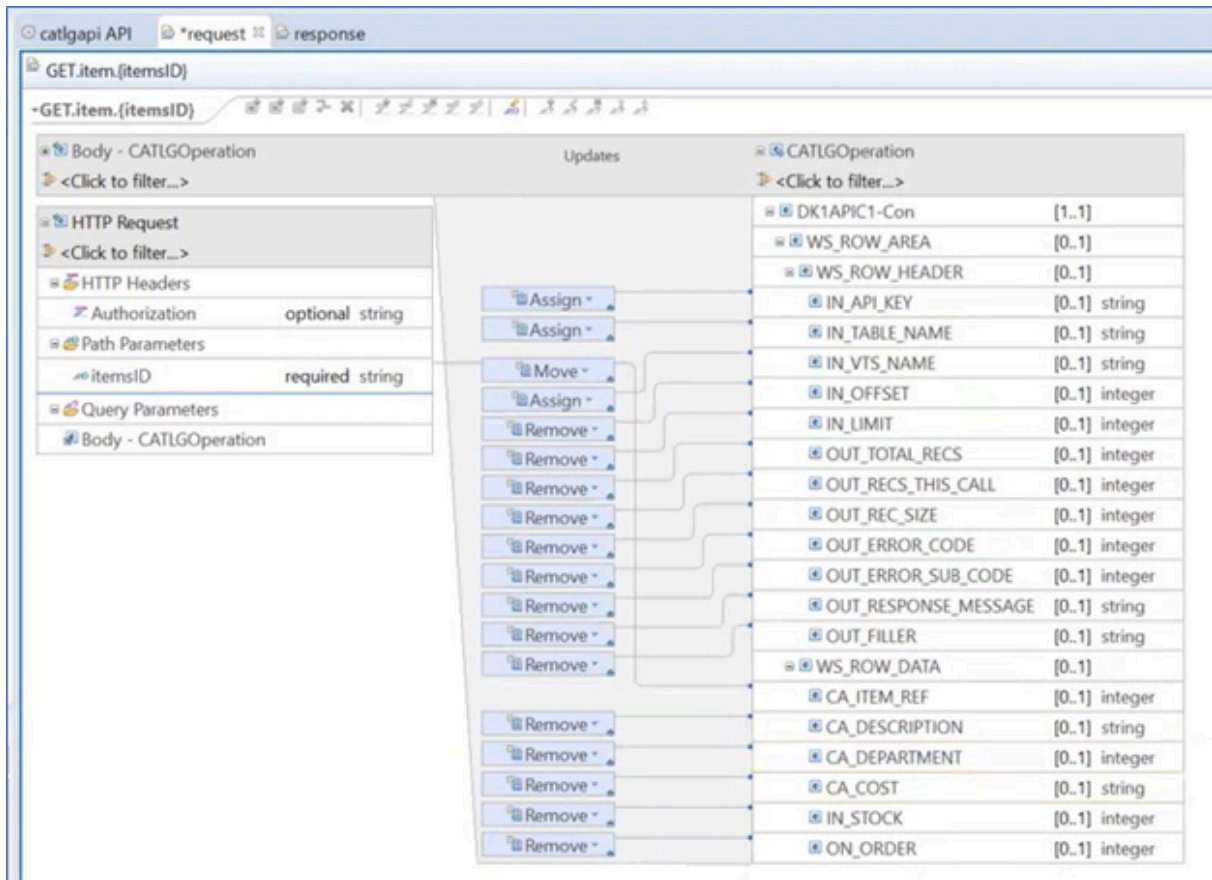


Figure 3-34: Request Mapping

Note: The `IN_TABLE_NAME` and `IN_VTS_NAME` fields can also be supplied as query parameters.

Mapping the Response

1. Next to the Request tab, select the **Response** tab.
2. Expand the `DK1APIC1-Con` to display the whole copybook.
3. Select the following fields in the **WS_ROW_HEADER** section:
 - `IN_API_KEY`
 - `IN_TABLE_NAME`
 - `IN_VTS_NAME`
 - `IN_OFFSET`
 - `IN_LIMIT`
 - `OUT_TOTAL_RECS`
 - `OUT_RECS_THIS_CALL`
 - `OUT_REC_SIZE`

- OUT_FILLER
4. Right-click to select [Add Remove transform](#).

Updates		Body - CATLGOperationResponse
		<Click to filter...>
		DK1APIC1-Con [1..1]
		WS_ROW_AREA [0..1]
		WS_ROW_HEADER [0..1]
Remove		IN_API_KEY [0..1] string
Remove		IN_TABLE_NAME [0..1] string
Remove		IN_VTS_NAME [0..1] string
Remove		IN_OFFSET [0..1] integer
Remove		IN_LIMIT [0..1] integer
Remove		OUT_TOTAL_RECS [0..1] integer
Remove		OUT_RECS_THIS_CALL [0..1] integer
Remove		OUT_REC_SIZE [0..1] integer
		OUT_ERROR_CODE [0..1] integer
		OUT_ERROR_SUB_CODE [0..1] integer
		OUT_RESPONSE_MESSAGE [0..1] string
Remove		OUT_FILLER [0..1] string
		WS_ROW_DATA [0..1]
		CA_ITEM_REF [0..1] integer
		CA_DESCRIPTION [0..1] string
		CA_DEPARTMENT [0..1] integer
		CA_COST [0..1] string
		IN_STOCK [0..1] integer
		ON_ORDER [0..1] integer

Figure 3-35: Response Mapping

Deploy API to Server

1. In the Project Explorer section, right-click and select **CATLGAPI > z/OS Connect EE > Deploy API to z/OS Connect EE Server**. The [Deploy API window](#) will display.

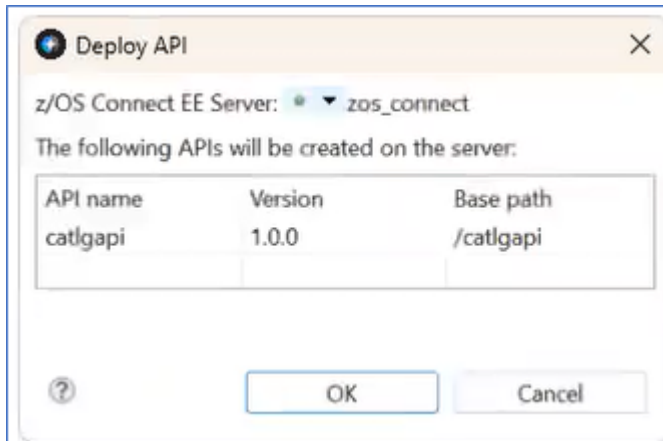


Figure 3-36: Deploy API window

2. Select the OK button. The [Deploy API window](#) will close. The [Deploy API to z/OS Connect EE Server Result window](#) will display.

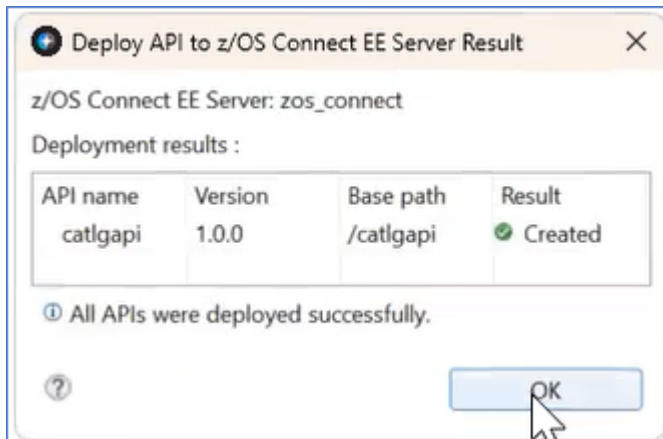


Figure 3-37: Deploy API to z/OS Connect EE Server Result window

3. Select the OK button. The [Deploy API to z/OS Connect EE Server Result window](#) will close.
4. In the [z/OS Connect EE Servers](#) panel the **catlgapi** displays.

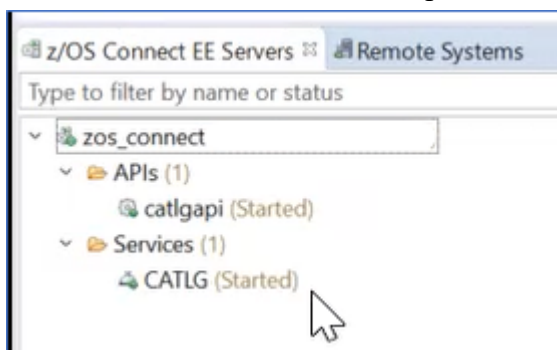


Figure 3-38: catlgapi started

Chapter

4

Designing APIs for Multiple Record Retrieval

This Chapter explains how to design and customize APIs using IBM Explorer for z/OS to create services and REST APIs for accessing z/OS resources.

The steps include:

1. [Create a Service Project.](#)
2. [Deploy the Service Project.](#)
3. [Create and configure an API Project.](#)
4. [Map GET Method for the request to the API.](#)
5. [Map GET Method for the response from the API.](#)
6. [Deploy the API Project.](#)

Note: The names of all tableBASEtables and VTSeS in this documentation are for test purposes. Please use your organization's naming schema.

Prerequisites for Multiple Responses

z/OS Connect V3.0 or later is installed.

IPIC connection to CICS has been configured.

Data is loaded into tableBASE table using a VTS-TSR with an index that can retrieve multiple responses.

Customized corresponding copybooks are completed.

Note: The following steps create an operation for multiple record retrieval and multiple record retrieval with a hexadecimal response. For additional information about multiple record retrieval refer to [IBM z/OS Connect](#) topic about how to **Create an operation to inquire about multiple items.**

Create a New Service Project for Multiple Records

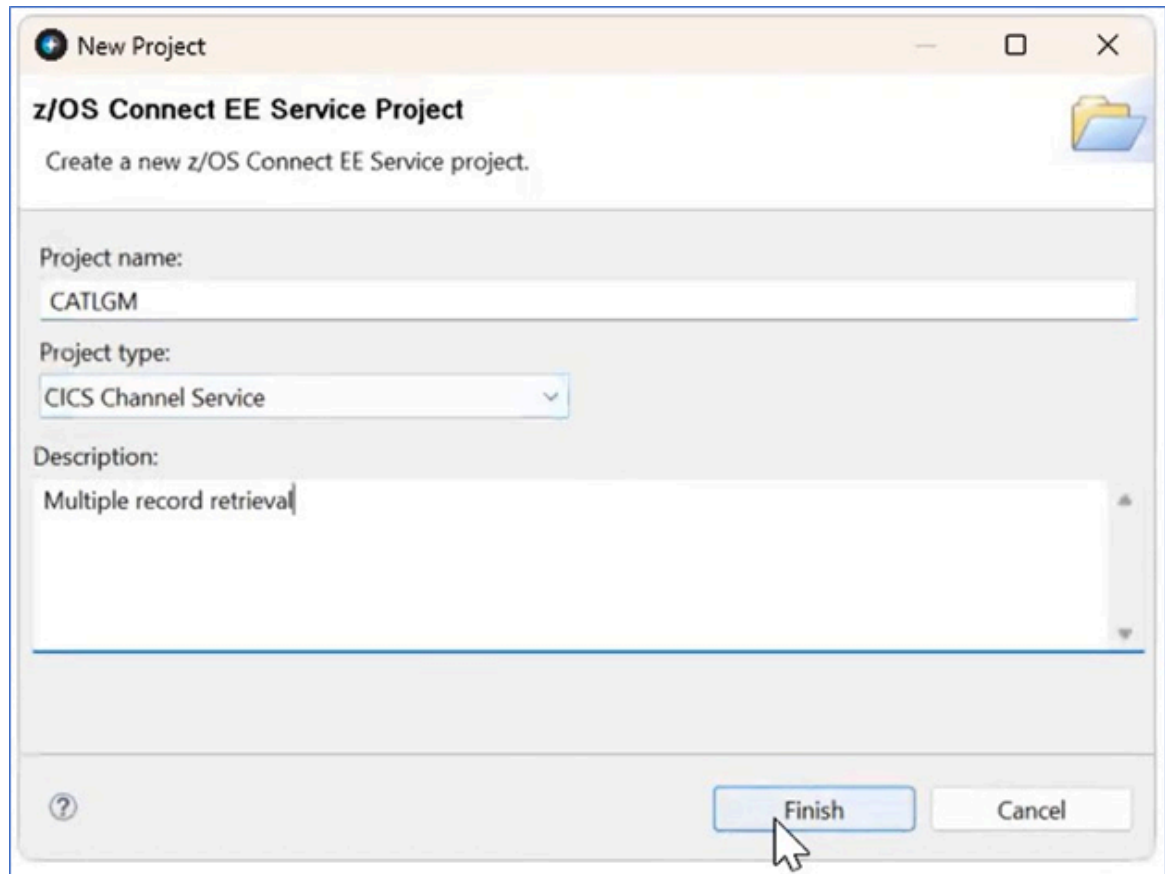
Creating the service and defining the array to provide the count of records returned along with their data.

1. Select **File > New > z/OS Connect EE Service Project**. The [New Project window](#) will open.
2. Enter the following details:

- a. Project Name is **CATLGM**

Note: CATLGM is the name provided for this sample. Please use your organization's naming schema.

- b. Project Type is **CICS Channel Service**.
c. Enter an optional description.



The screenshot shows a 'New Project' dialog box with the title 'z/OS Connect EE Service Project'. Below the title is the instruction 'Create a new z/OS Connect EE Service project.' The dialog contains three input fields: 'Project name:' with the text 'CATLGM', 'Project type:' with a dropdown menu showing 'CICS Channel Service', and 'Description:' with the text 'Multiple record retrieval'. At the bottom right, there are two buttons: 'Finish' and 'Cancel'. A mouse cursor is pointing at the 'Finish' button.

Figure 4-1: Create a new Service Project

- d. Select the **Finish** button. The [New Project Window](#) will close.

- On the [Configuration tab](#), enter the **Connection Reference** to the CICS region.

Service Project Editor: Configuration

Required Configuration

Enter the required configuration for this service.

Coded character set identifier (CCSID): 37

Connection reference: Conn

Optional Configuration

Enter the optional configuration for this service.

Transaction ID:

Transaction ID usage:

Bidi configuration reference:

Use context containers: ☐

Context containers HTTP headers:

Add another

Figure 4-2: CICSConn Configuration

Create a New Service Request for Multiple Records

- Select the [Service Project Editor: Definition tab](#). The CATLG Service will display. In the Program field, enter Program name DK1API01.

Program

Program: DK1API01

Define Request and Response Service Interfaces

Create new request and response service interfaces or select existing ones.

Create Service Interface... Import Service Interface...

Request service interface: Edit

Response service interface: Edit

Set advanced data conversion options Advanced Options...

Figure 4-3: Service Project Editor Definition tab

- In the [Define Request and Response Service Interfaces](#) section:

- a. Select the **Create Service Interface** button. The [Create Service Interface window](#) will display.

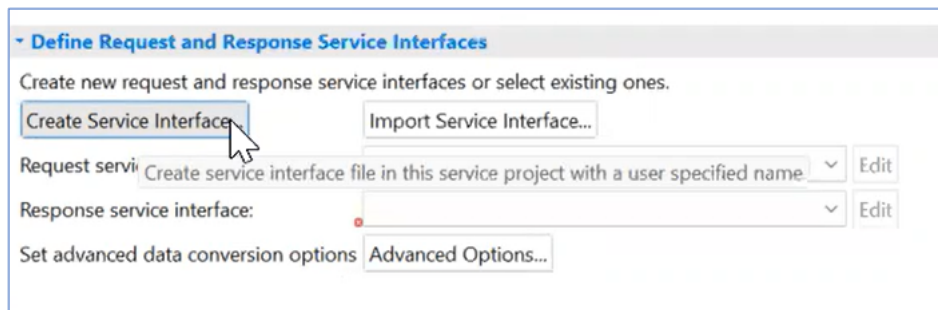


Figure 4-4: Define the Service Request

- b. Enter the Service Interface name **CATLG**. Select the OK button. The [Create Service Interface window](#) will close. A CATLG tab will display in the [Service Interface Editor](#).

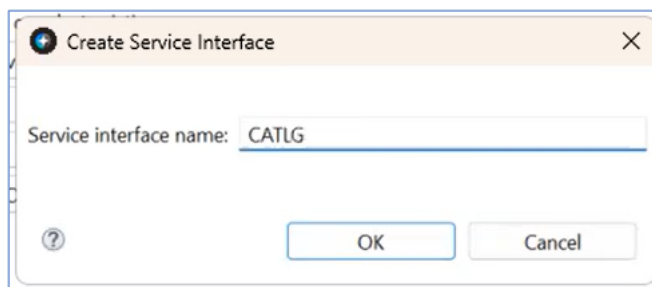


Figure 4-5: Create Service Interface Window

3. In the [Service Interface editor](#), right-click to select the **Container 1**. Select **Edit Container**. The [Edit Container window](#) will display.

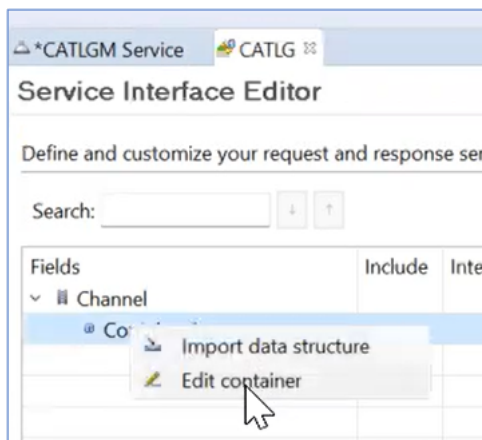


Figure 4-6: CATLG Service Interface editor

4. In the Container name field enter DK1APIC1-Con. Select the OK button. The [Edit Container window](#) will close.

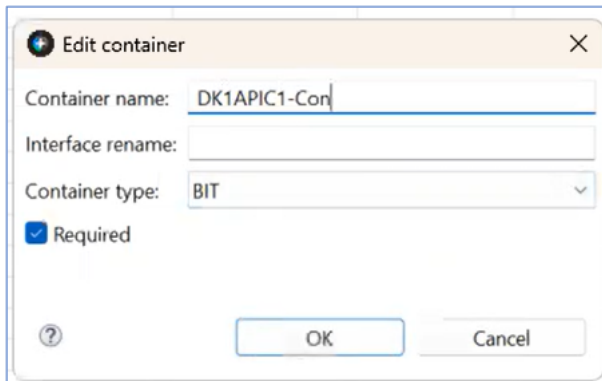


Figure 4-7: Edit Container Window

5. On the [Service Interface Editor](#) right-click to select DK1APIC1-Con. Select **Import Data Structure**. The [Import window](#) will display.

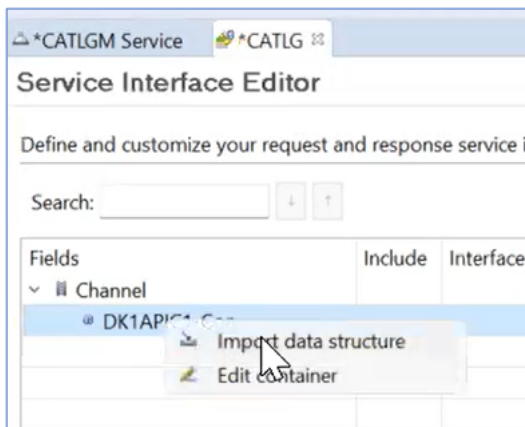


Figure 4-8: CATLG Import Data Structure

6. In the [Import Window](#), select the Source PDS.

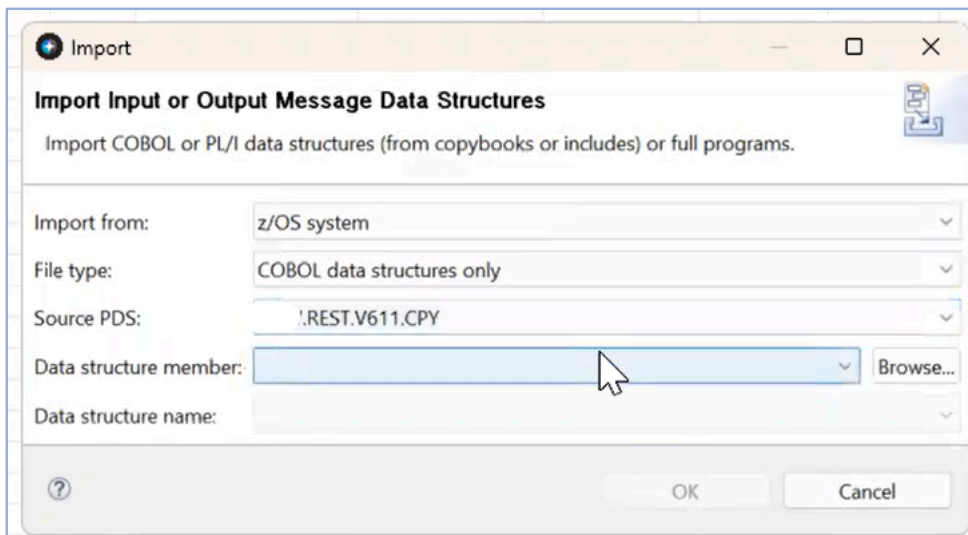


Figure 4-9: Source PDS Import Request

7. Select the **Browse** Button. The [Import Application Data Structure Source window](#) will display.

- a. Enter the **Source PDS** and <Enter>. The Matching Data Sets field will populate with the members of the PDS entered.
- b. Expand and select the copybook member **CATLG**. Select the Finish button. The [Import Application Data Structure Source window](#) will close.

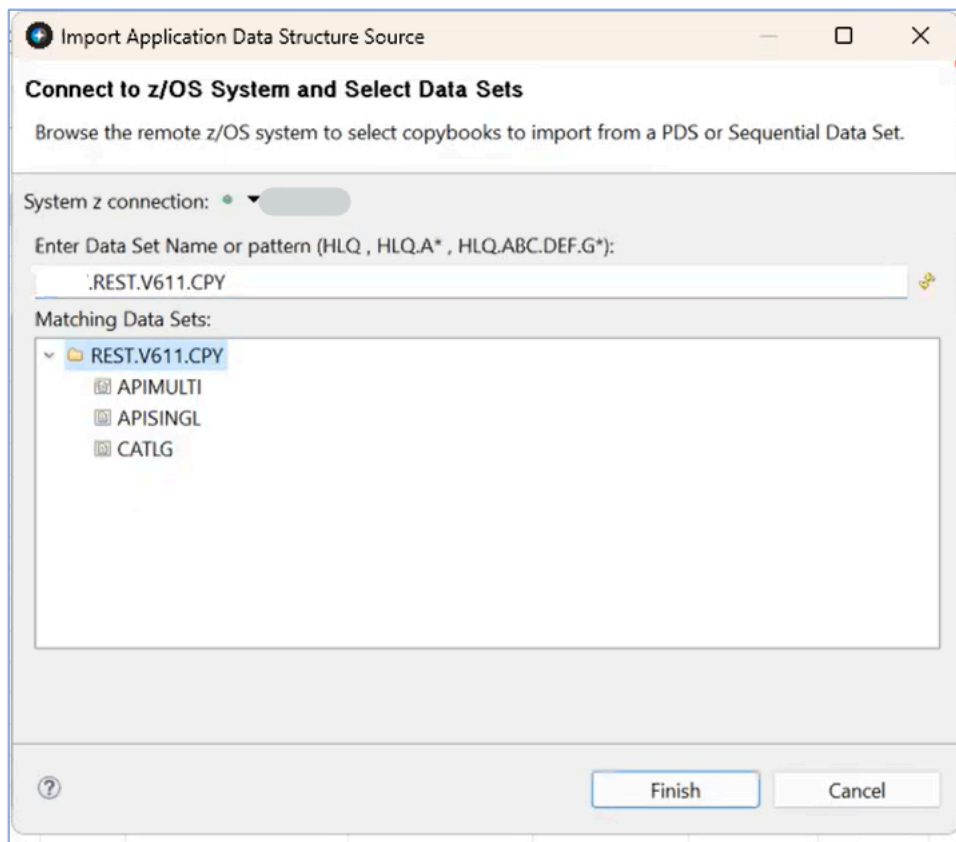


Figure 4-10: CATLG Import Application Data Structure Source window

8. In the Define Request and Response Service Interfaces section of the Definition panel, select [Request service interface](#) field select **CATLG.si**.

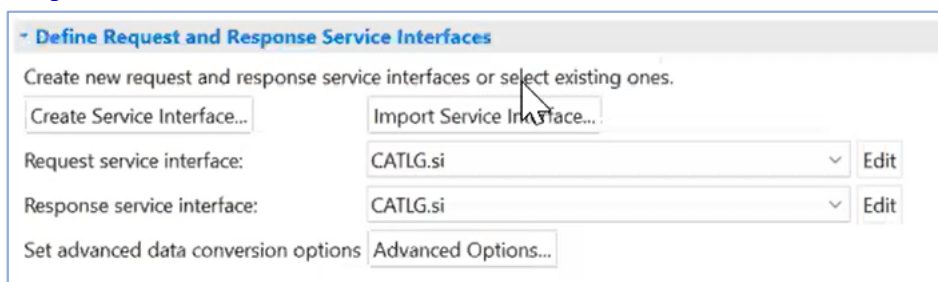


Figure 4-11: CATLG Request Service Interface

Create new Service Response for Multiple Records

For multiple responses, the request is mapped to the single copybook, CATLG. The Response is mapped to the multiple copybook, CATLGM.

The difference between CATLG and CATLGM is the array in the WS.ROW.DATA field of the copybook. It can retrieve a maximum of 100 records. The Array in CATLGM must be defined to

return the proper count of records. This tells the Response the number of records to account for in the array.

1. Select the [Create Service Interface](#) button. The [Create Service Interface window](#) will display.

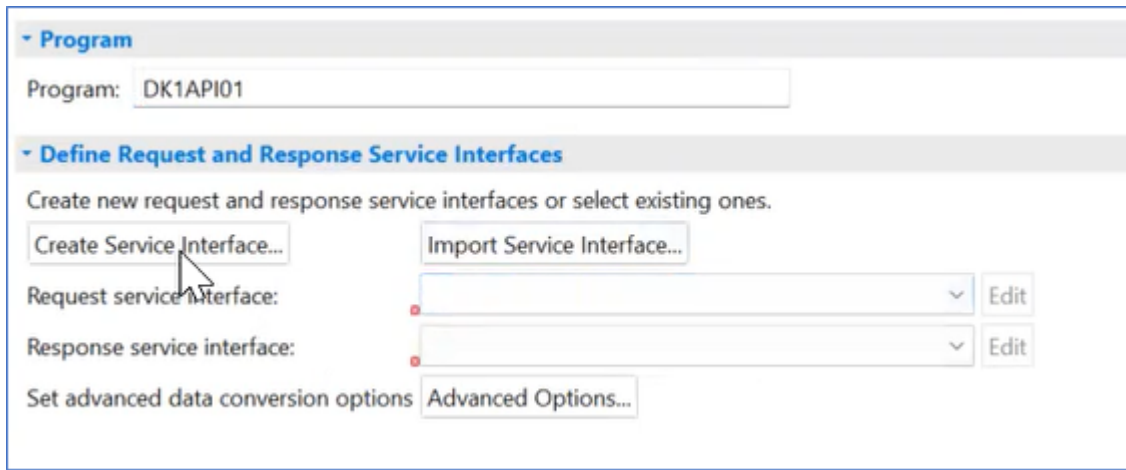


Figure 4-12: Create Service Interface Button

2. Enter the Service Interface name **CATLGM**. Select the OK button. The [Create Service Interface Window](#) will close. A CATLGM tab will display the [Service Interface Editor](#).

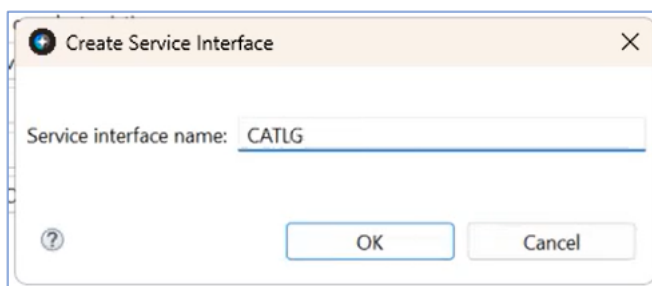


Figure 4-13: Create Service Interface Window

3. In the [Service Interface editor](#), right-click to select the **Container 1** and select **Edit Container**. The [Edit Container window](#) will display.

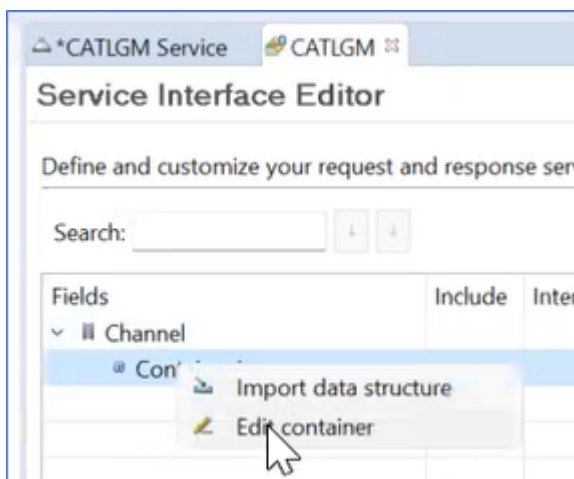


Figure 4-14: Edit Container

4. In the Container Name field enter DK1APIC1-Con. Select the OK button. The [Edit Container window](#) will close.

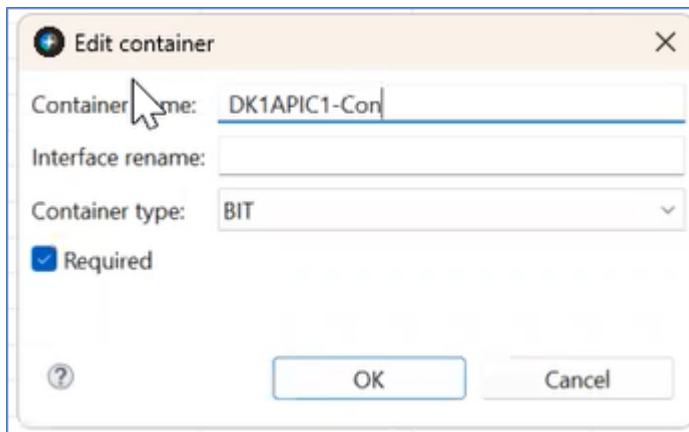


Figure 4-15: Edit Container Window

5. Right-click to select DK1APIC1-Con and select [Import Data Structure](#). The [Import Window](#) will display.

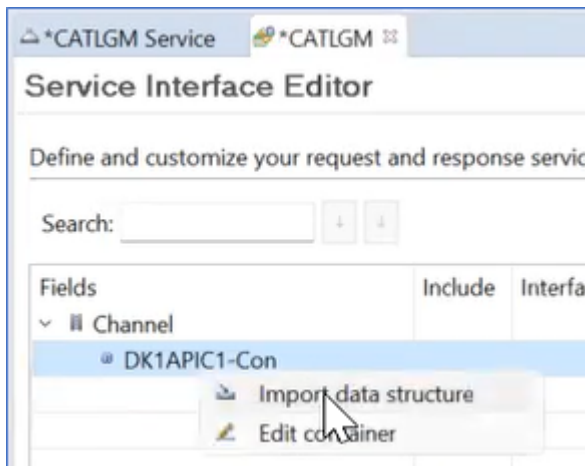


Figure 4-16: Import Structure

6. In the [Import Window](#), select the **Source PDS**.

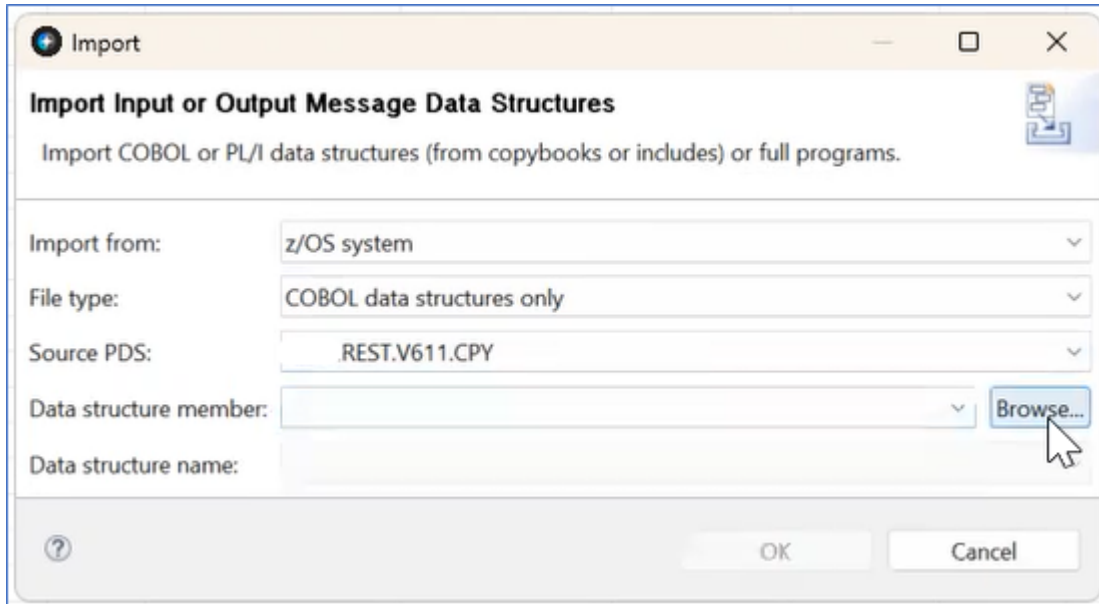


Figure 4-17: Import Window for Source PDS

7. Select the **Browse** Button. [The Import Application Data Structure Source window](#) will display.
 - a. Input the **Source PDS** and **<Enter>**. The Matching Data Sets field will populate with the members of the PDS entered.

- b. Expand the Matching Data Sets and select the copybook member **CATLGM**. Select the Finish button. The [Import Application Data Structure Window](#) will close.

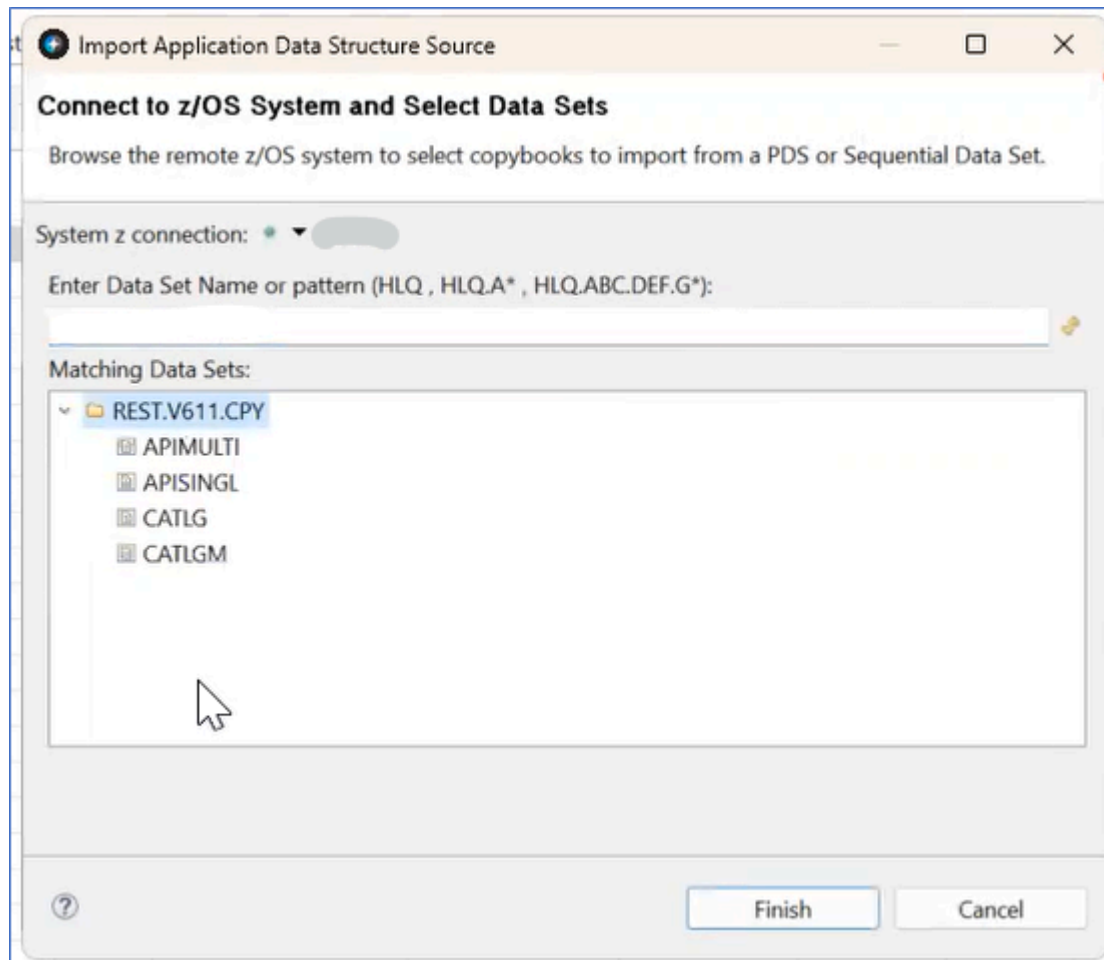


Figure 4-18: Import Application Data Structure Source Window

8. Define the Array.

- a. Right-click to select the **WS_ROW_DATA** field select **Edit Field**. The **Edit Field window** will display.

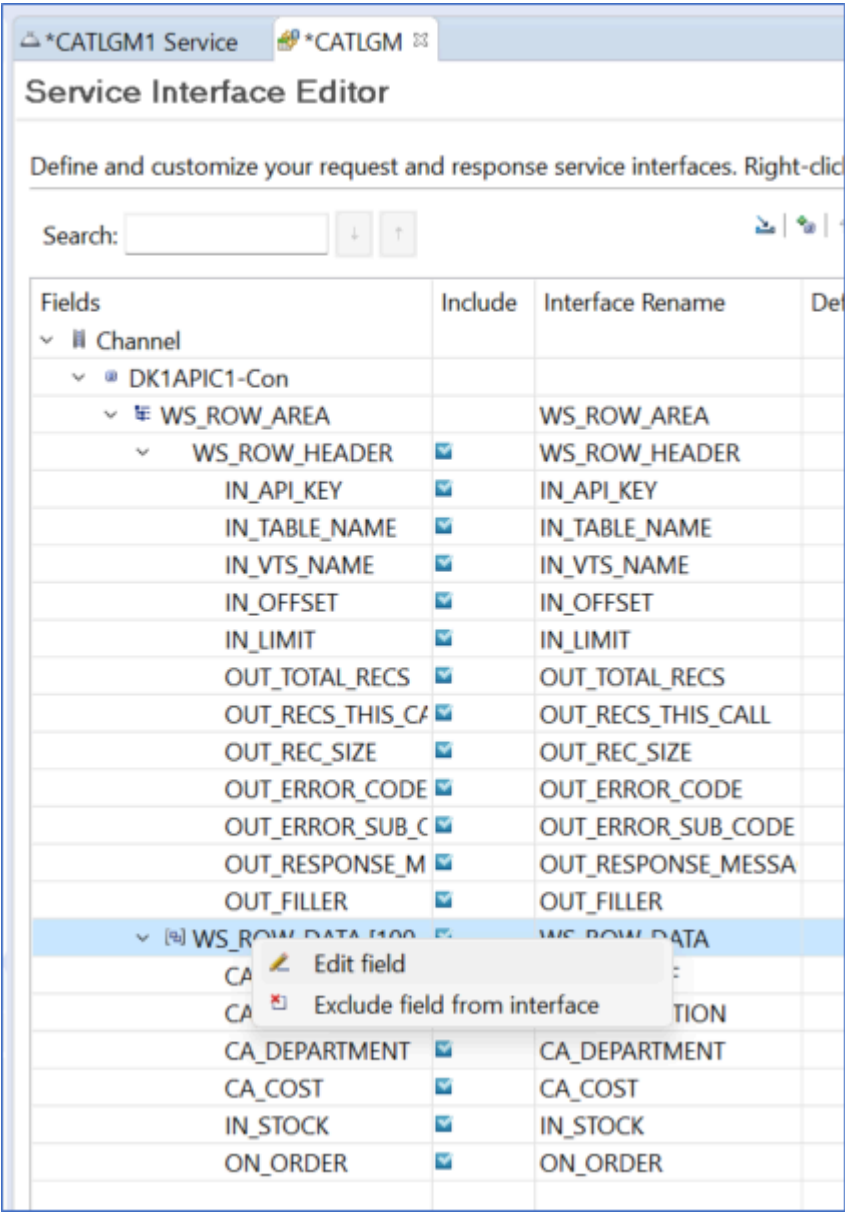
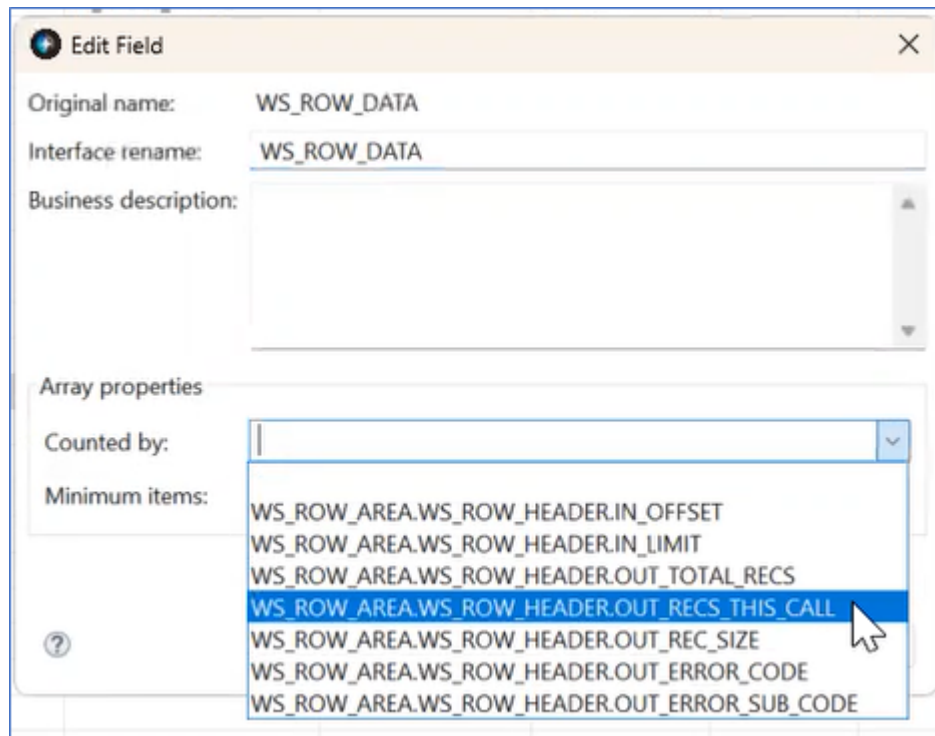


Figure 4-19: Disable OUT_RECS_THIS_CALL

- b. To provide the [Array Definition](#), in the Counted By field, select **WS_ROW_AREA.WS_ROW_HEADER.OUT_RECS_THIS_CALL**.



The screenshot shows the 'Edit Field' dialog box with the following fields:

- Original name: WS_ROW_DATA
- Interface rename: WS_ROW_DATA
- Business description: (empty text area)
- Array properties section:
 - Counted by: (dropdown menu)
 - Minimum items: (empty text area)

The dropdown menu for 'Counted by' is open, showing the following options:

- WS_ROW_AREA.WS_ROW_HEADER.IN_OFFSET
- WS_ROW_AREA.WS_ROW_HEADER.IN_LIMIT
- WS_ROW_AREA.WS_ROW_HEADER.OUT_TOTAL_RECS
- WS_ROW_AREA.WS_ROW_HEADER.OUT_RECS_THIS_CALL** (highlighted)
- WS_ROW_AREA.WS_ROW_HEADER.OUT_REC_SIZE
- WS_ROW_AREA.WS_ROW_HEADER.OUT_ERROR_CODE
- WS_ROW_AREA.WS_ROW_HEADER.OUT_ERROR_SUB_CODE

Figure 4-20: Array Definition

- c. Select the OK button. The [Edit Field window](#) will close. The [OUT_RECS_THIS_CALL](#) field will be disabled. There is a red X next to the OUT_RECS_THIS_CALL field.

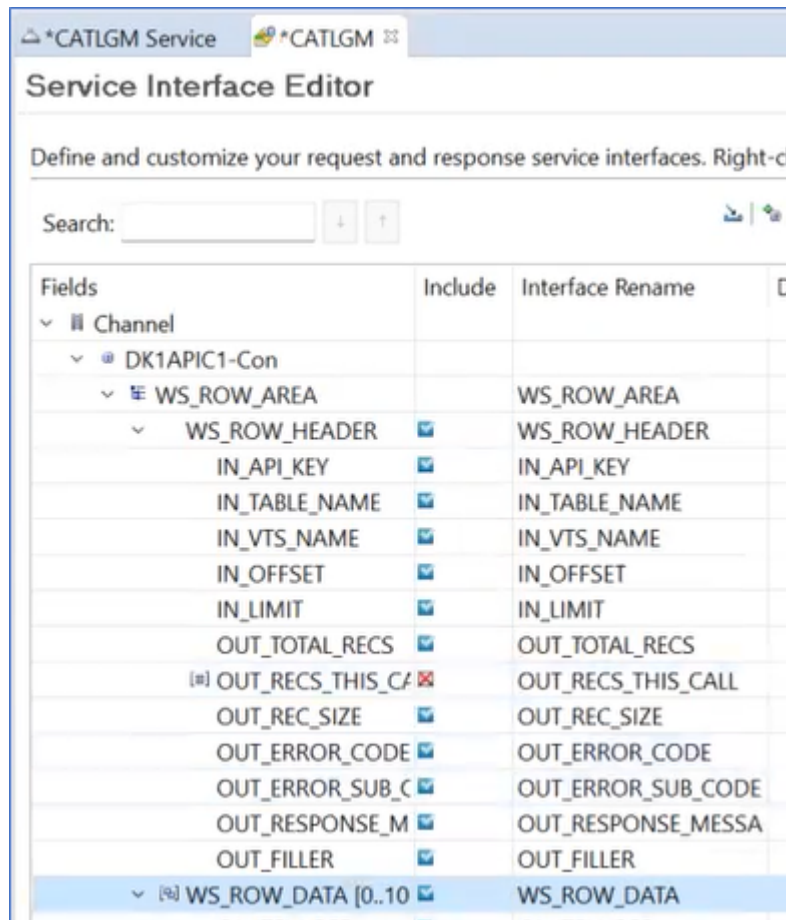


Figure 4-21: OUT_RECS_THIS_CALL disabled

9. In the [Response service interface](#) field select the CATLGM.si .

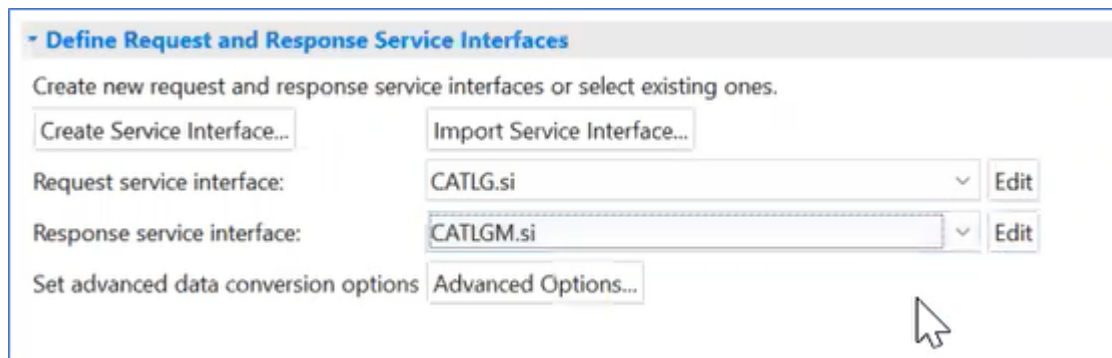


Figure 4-22: Response Service Interface

Deploy the Service Project

To deploy the service:

1. Right-click the **CATLGM** service to select **z/OS Connect EE > Deploy Service to z/OS Connect EE Server**. The [Deploy Service window](#) will display.

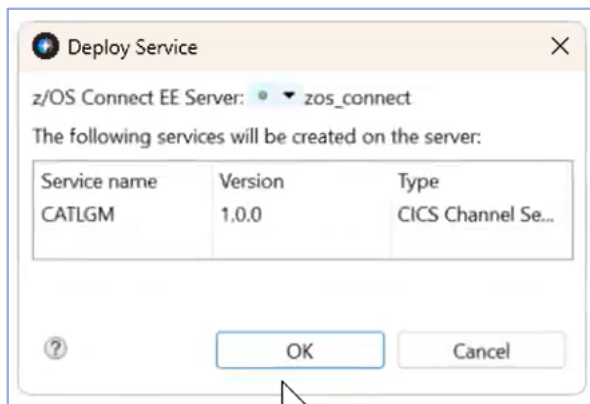


Figure 4-23: CATLGM Deploy Service

2. Select the OK button. The [Deploy Service window](#) will close. The [Deploy Service to z/OS Connect EE Server Result Window](#) will display.

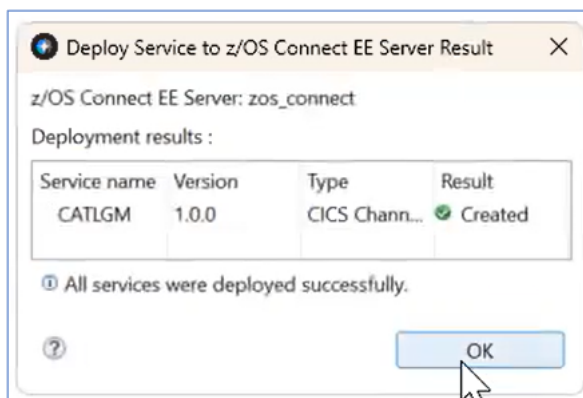


Figure 4-24: Deploy Server to z/OS Connect EE Server Result Window

3. Select the OK button. The [Deploy Service to z/OS Connect EE Server Result Window](#) will close. The [CATLGM Deployed Service](#) will display in the z/OS Connect EE Servers panel as Started.

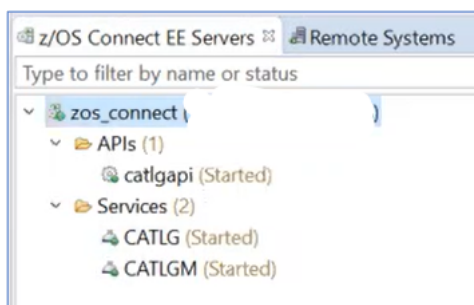


Figure 4-25: CATLGM Deployed

Create a New API Project for Multiple Records

1. Select **File > New > z/OS Connect EE API Project**. The [New Project window](#) will display.

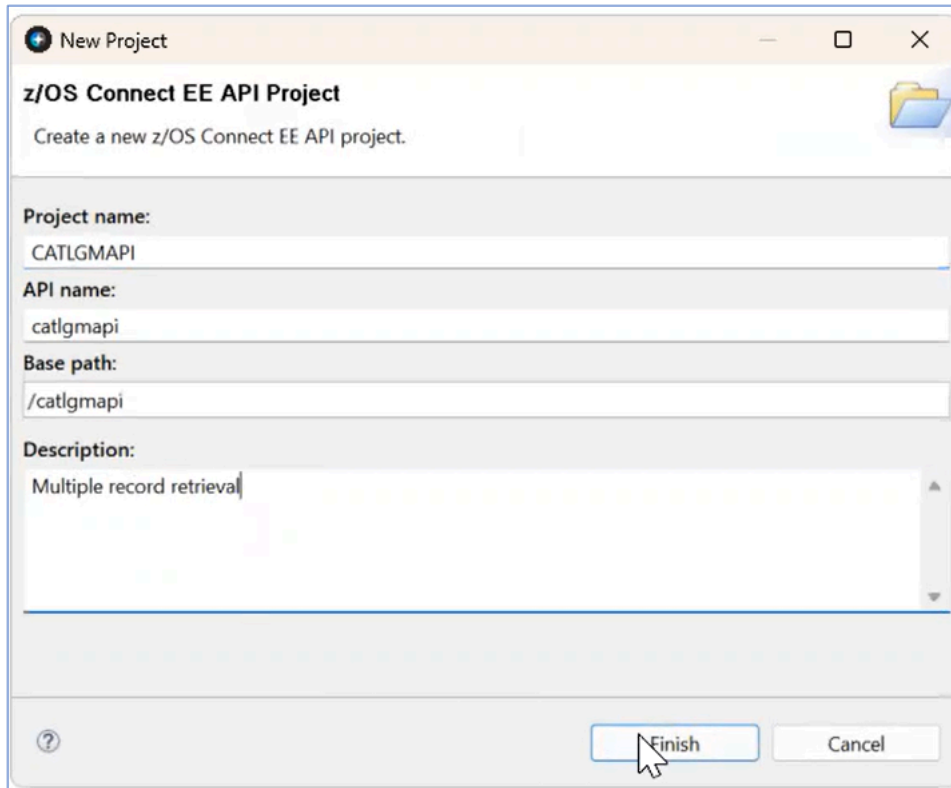


Figure 4-26: New API Project

2. In the [New Project window](#) enter the following information:
 - a. The Project name is **CATLGMAPI**.
 - b. the API Name will populate with **catlgmapi**. It is an editable field.
 - c. The Base Path will populate with **/catlgmapi**. It is an editable field.
 - d. Enter an optional description.
 - e. Select the **Finish** button. The [New Project window](#) will close and the [API Editor](#) will display.

3. Change the path name to `/items/{item-desc}`.

The screenshot shows the 'API Editor' window for 'catlgmapi API'. The 'Describe your API' section contains the following fields:

- Name: catlgmapi
- Description: Multiple record retrieval
- Base path: /catlgmapi
- Version: 1.0.0

The 'Contact Information' section is empty. The 'Path' section shows the path `/items/{item-desc}`. The 'Methods (4)' section lists four methods: POST, GET, PUT, and DELETE. Each method has a 'Service...' button, a 'Mapping...' button, and a red X icon for deletion.

Figure 4-27: catlgmapi Editor

4. Remove the **POST**, **PUT**, and **DELETE** methods by selecting the box with a red **X** to the right of the Method. A window will display to confirm the deletion. The **GET Method** should be the only Method.

The screenshot shows the 'API Editor' window for 'catlgmapi API' after deleting the POST, PUT, and DELETE methods. The 'Path' section shows the path `/items/{item-desc}`. The 'Methods (1)' section lists one method: GET. The GET method has a 'Service...' button, a 'Mapping...' button, and a red X icon for deletion.

Figure 4-28: GET Method

5. In the Project Explorer window, right-click to select the **CATLGM**. Select **z/OS Connect EE > Export z/OS Connect EE Service Archive**. The [Export Service Package Window](#) displays.

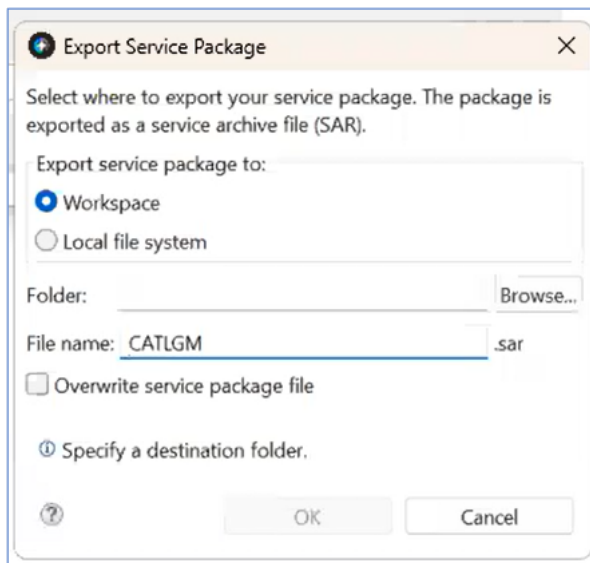


Figure 4-29: Export Service Package

6. Select the **Browse** button. The [Select Workspace Folder window](#) will display.

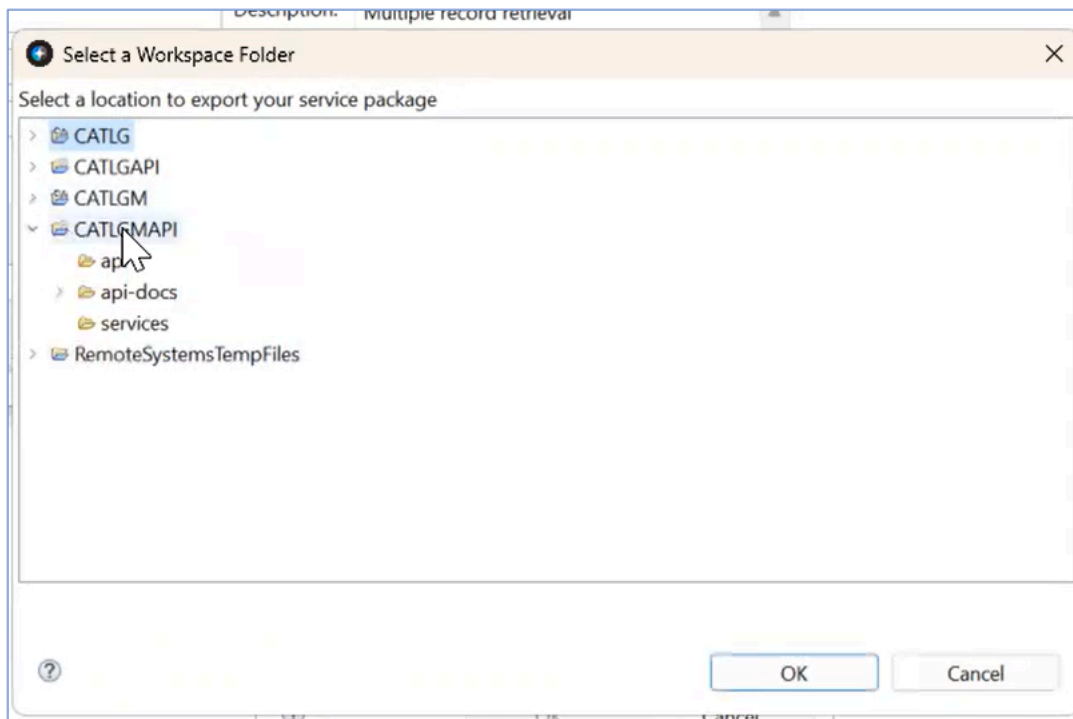


Figure 4-30: Select a Workspace Folder Window

- a. Select **CATLGMAPI**.
- b. Select the OK Button. The [Select a Workspace Folder Window](#) will close. The [Export Service Package Window](#) displays the Folder is mapped to /CATLGMAPI.

- c. The [Export Service Package window](#) displays the selected folder and file name. Select the OK button. The [Export Service Package window](#) will close.

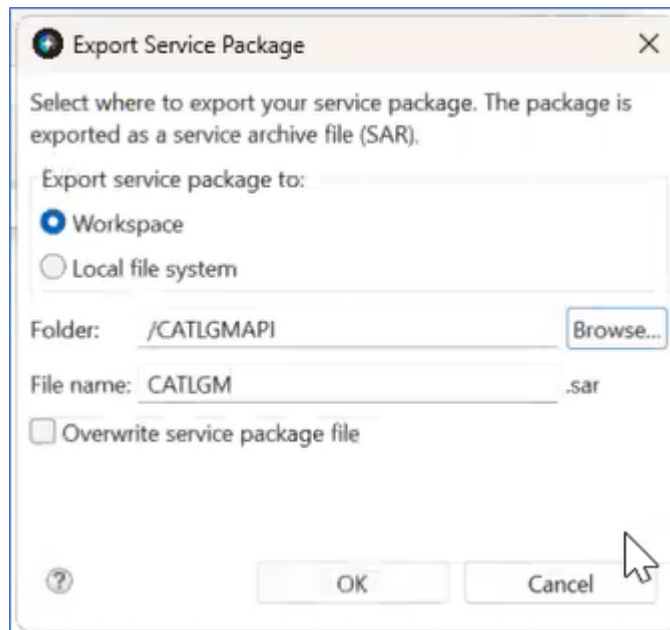


Figure 4-31: Completed Export Service Package

Mapping the GET Method

The following steps create an operation for multiple record retrieval from the EXMPCATA table. EXMPCATA has an alternate index using the product description as the key. For additional information about multiple record retrieval, refer to [IBM z/OS Connect](#) topic about how to **Create an operation to inquire about multiple items**.

Import the Service to the GET Method

1. Select the [Service](#) button next to the GET field. The [Select z/OS Connect EE Service Window](#) will display.

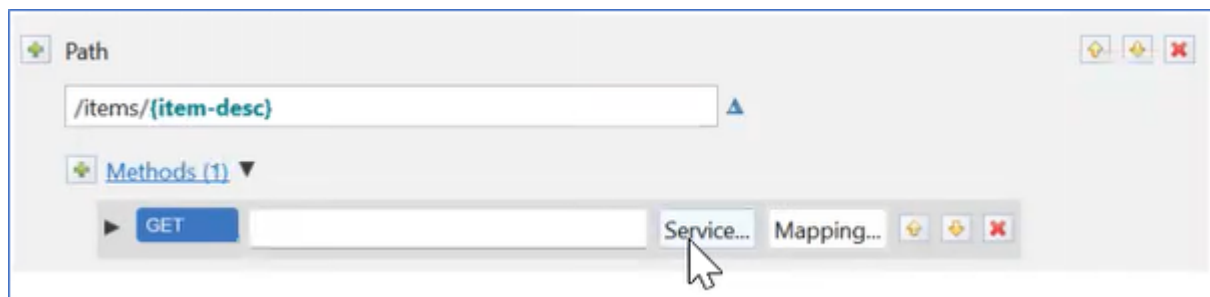


Figure 4-32: Service to GET

2. Select the **Workspace** button. The [Import z/OS Connect EE Services Window](#) will display.

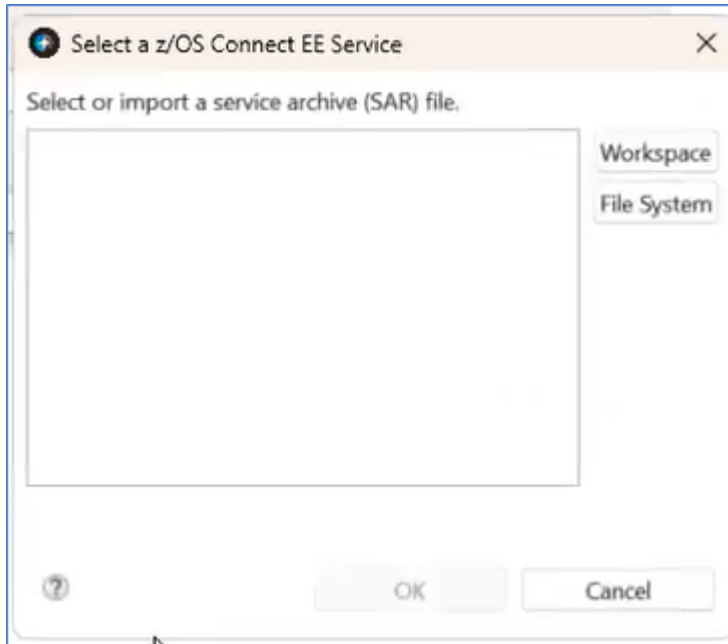


Figure 4-33: Select z/OS Connect EE Service Window

3. Expand **CATLGMAPI**.
 - a. Select **CATLGM.sar**.
 - b. Select the OK Button. The [Import z/OS Connect EE Services Window](#) will close. The [Import Services window](#) will display.

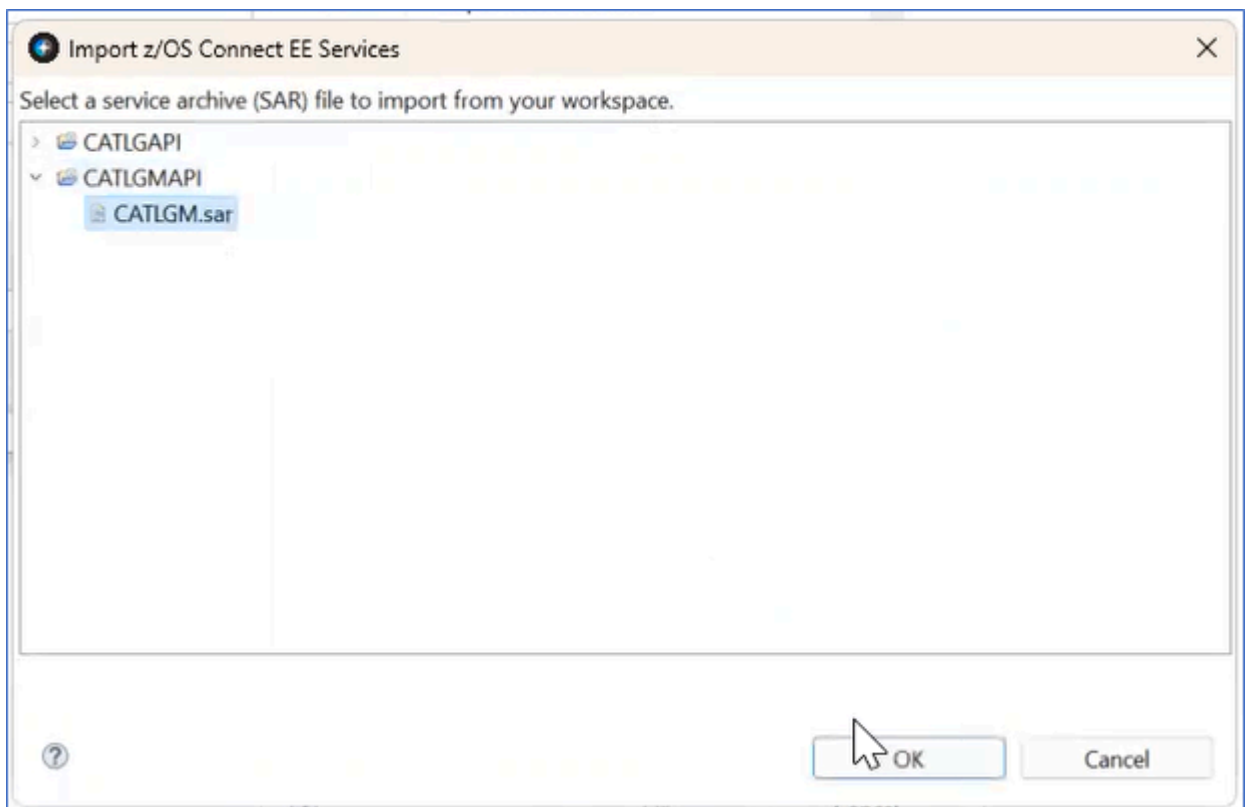


Figure 4-34: Import z/OS Connect EE Services Window

4. Select the OK button. The [Import Services window](#) will close.

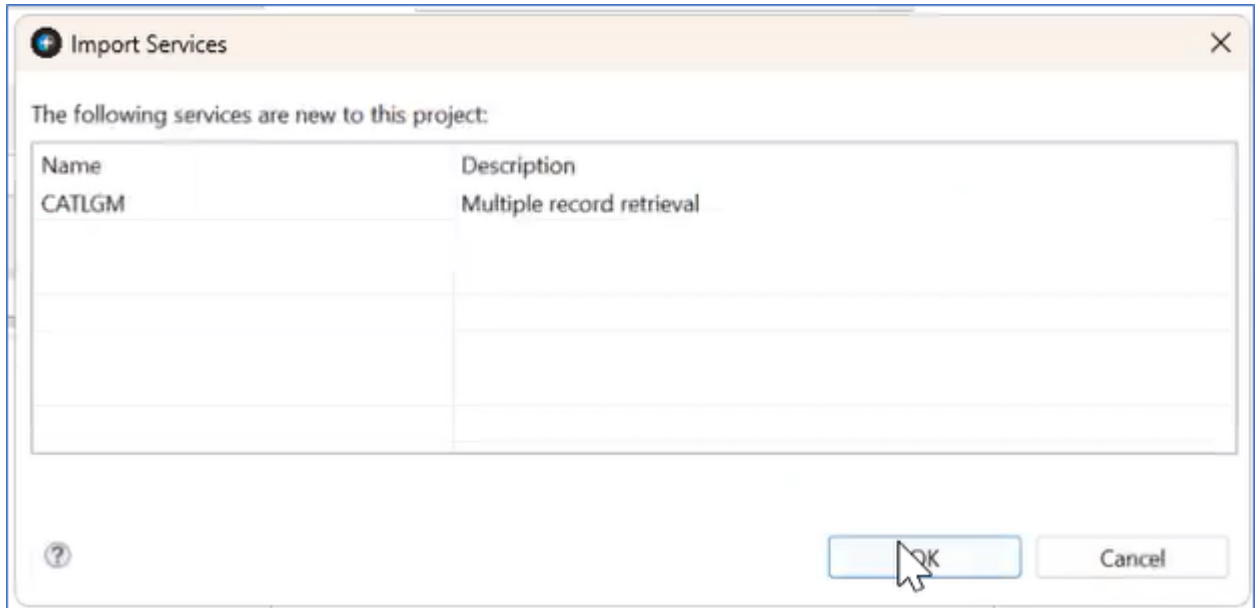


Figure 4-35: Import Services

Mapping the GET Request for Multiple Responses

Use the single copybook, CATLG, to map the request. One request will be made, so it will be based on the Product Description field. The Product Description will return multiple records. The table EXMPCATA uses an alternate index using the Product Description as the key.

1. In the API Editor select the [Mapping](#) button. Select **Open Both Mappings**. The [Request and Response](#) tabs will display.

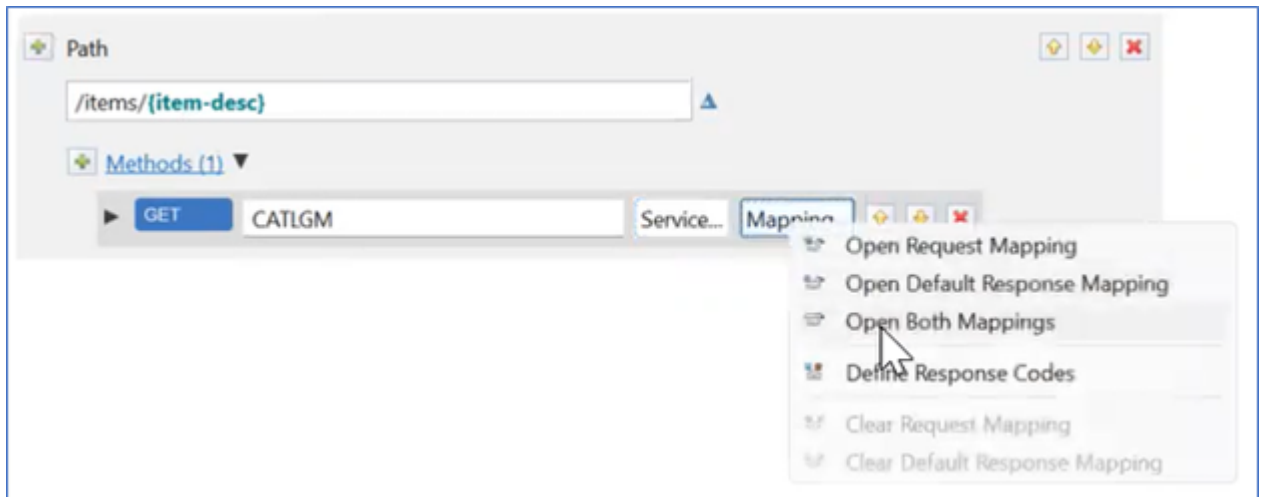


Figure 4-36: CATLGM Open both mappings

2. Select the **Request** tab.

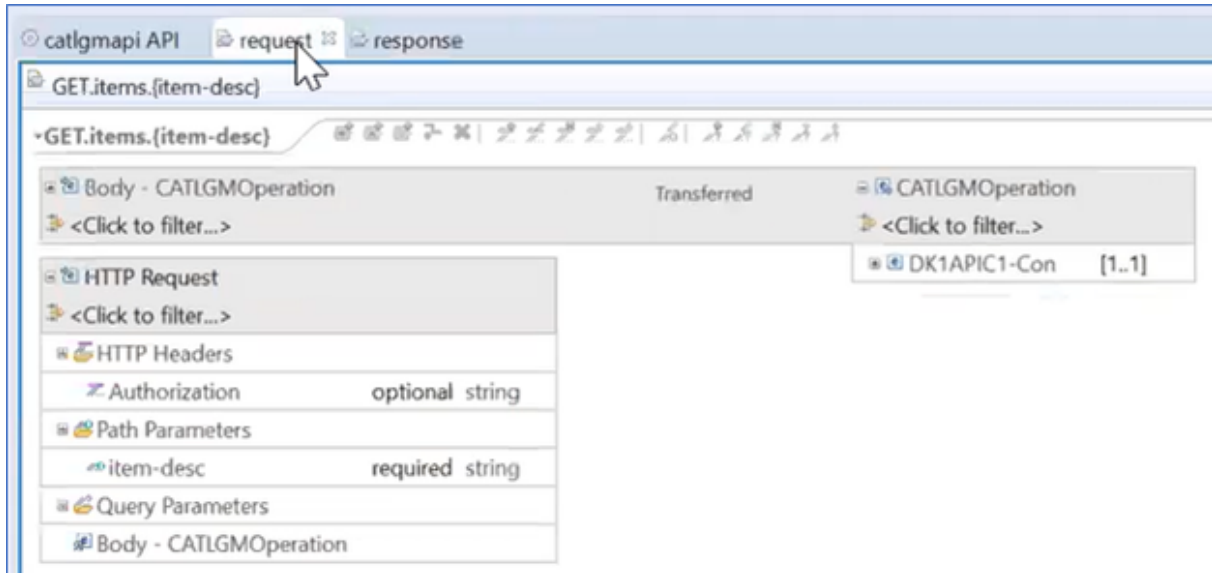


Figure 4-37: Request and Response Mapping

3. Expand **DK1APIC1-Con** to display the whole copybook.
4. Select the **IN_API_KEY** field in the copybook.

- a. Right-click to select [Add Assign transform](#).

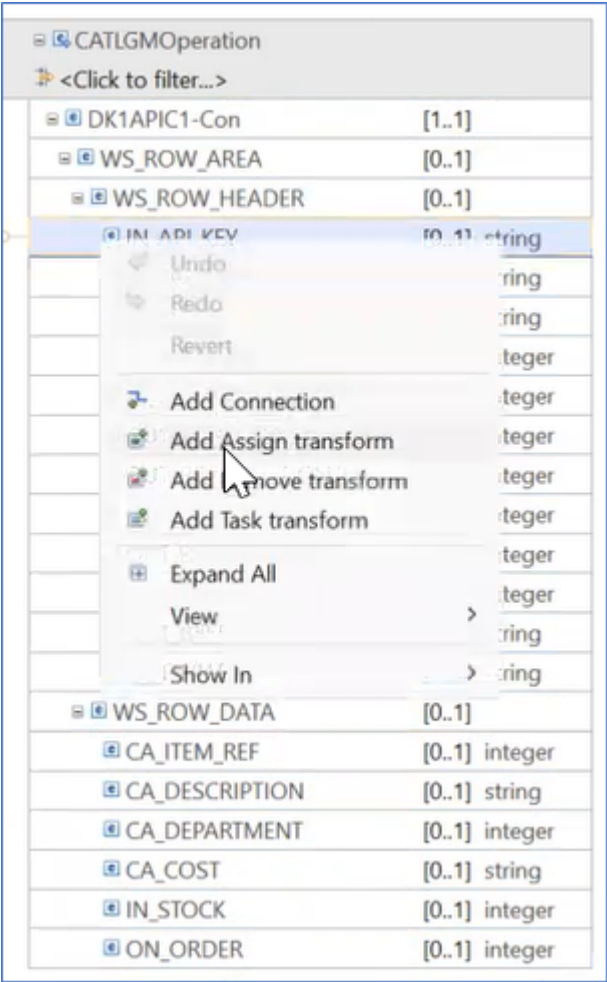


Figure 4-38: Add Assign Transform

- b. In the Properties Panel enter the value [API002](#). A box displays next to the field marking it as Assigned.

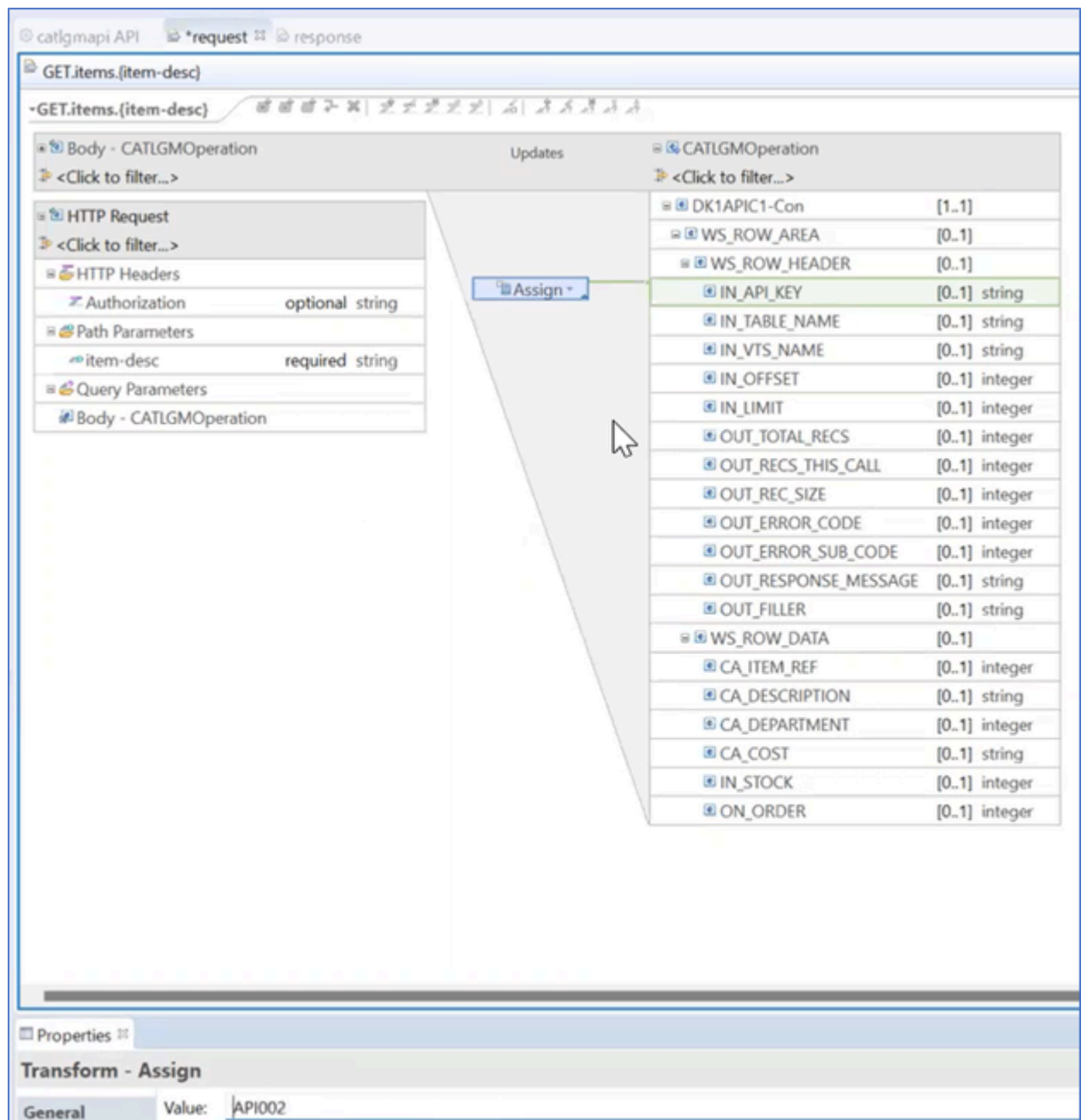


Figure 4-39: Assign IN_API_KEY

5. Select the [IN_TABLE_NAME](#) field in the copybook.
- a. Right-click to select **Add Assign transform**.

- b. In the Properties Panel enter the table name with an [alternate index](#) . A box displays next to the field marking it as Assigned.

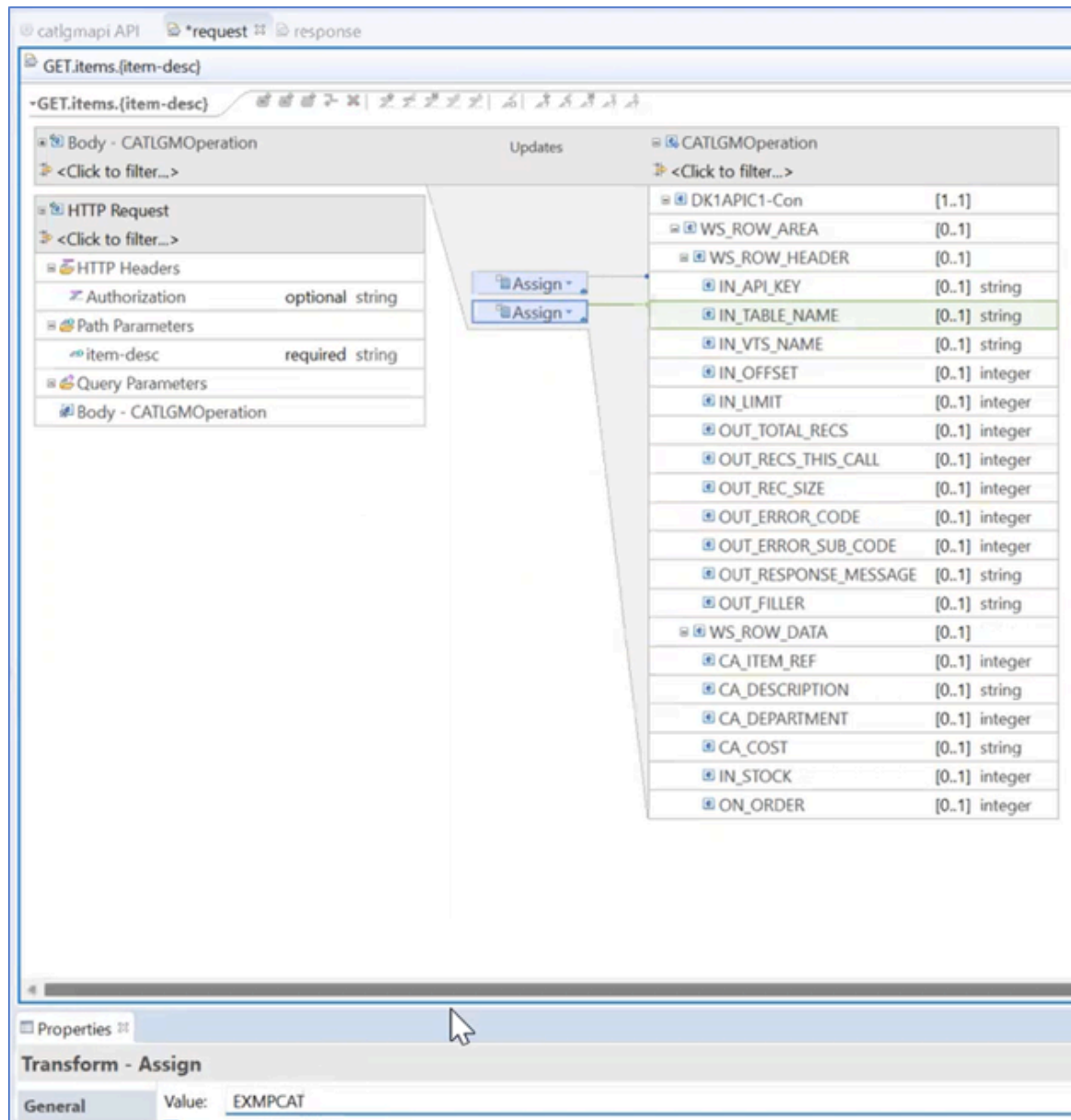


Figure 4-40: Assign IN_TABLE_NAME

6. Select the [IN_VTS_NAME](#) field in the copybook.
 - a. Right-click to select **Add Assign transform**.

- b. In the Properties Window enter the name of the VTS-TSR, **VTS2**. A box displays next to the field marking it as **Assigned**.

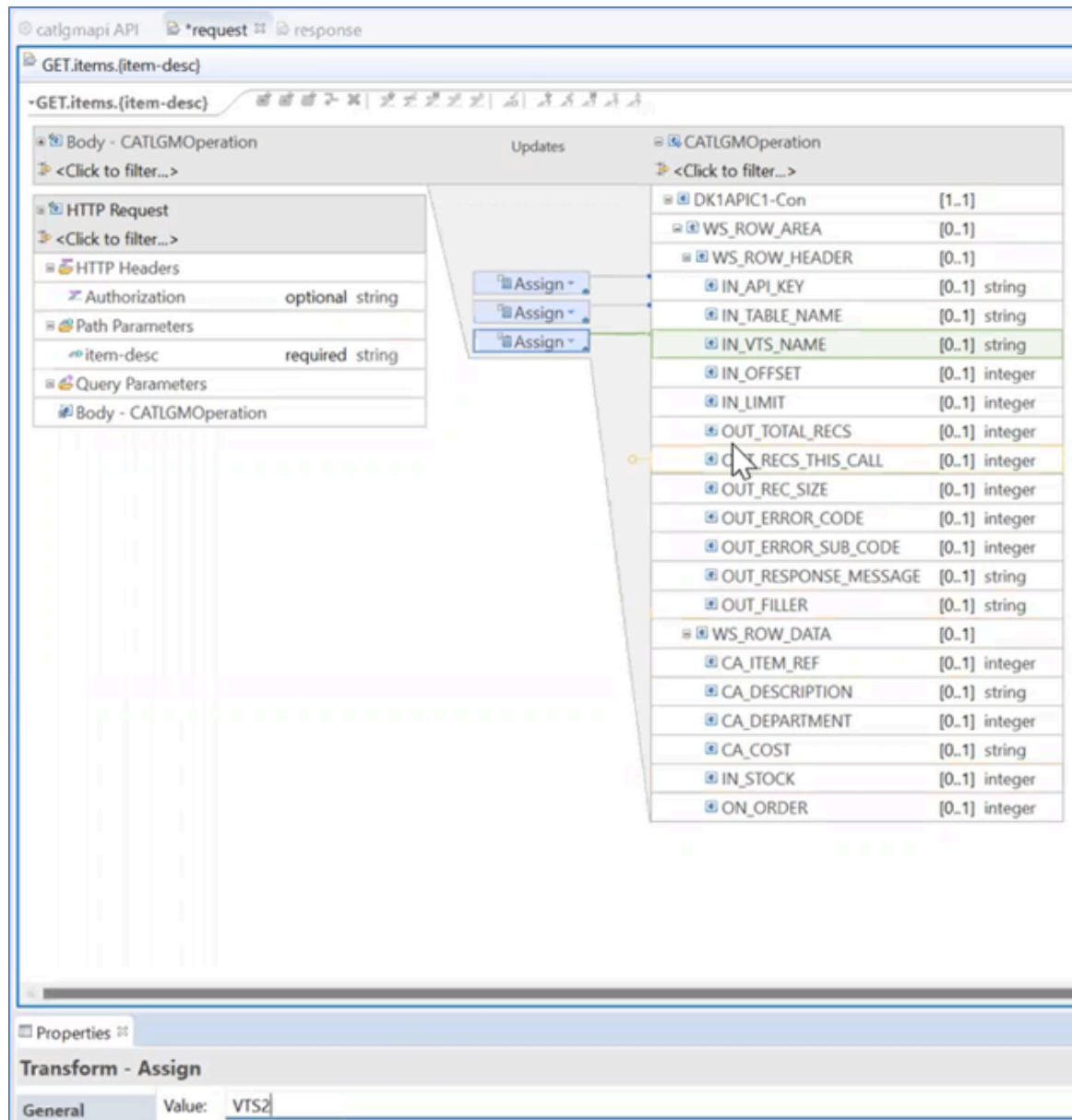


Figure 4-41: Assign VTS Name

7. Select the following fields in the WS_ROW_HEADER section:

- OUT_TOTAL_RECS
- OUT_RECS_THIS_CALL
- OUT_REC_SIZE
- OUT_ERROR_CODE
- OUT_ERROR_SUB_CODE
- OUT_RESPONSE_MESSAGE
- OUT_FILLER

8. Right-click to select **Add Remove Transform**. A box displays next to the field marking it as **Removed**.

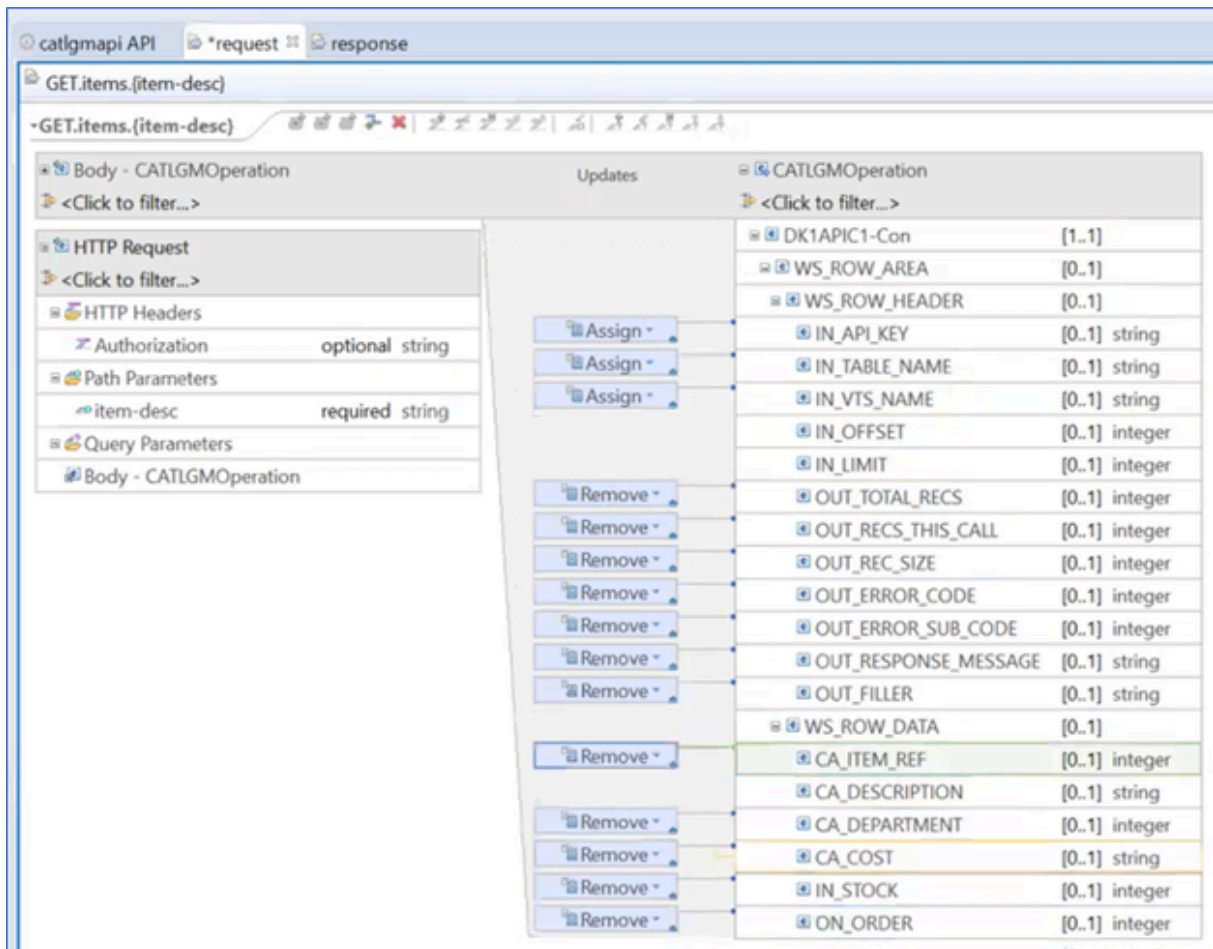


Figure 4-42: Request Add Remove Transform

9. Select the following fields associated with the request in the WS_ROW_DATA section:

- CA_ITEM_REF
- CA_DEPARTMENT
- CA_COST
- CA_STOCK
- IN_STOCK
- ON_ORDER

10. Select **Add Remove Transform**. A box displays next to each field marking it as **Removed**.

11. In the HTTP Request section, select **item-desc** listed as a Path Parameter. Drag it to the field in the WS_ROW_DATA section associated with the request, **CA_DESCRIPTION**. A box displays **Move** between the Path Parameter and the Request field.

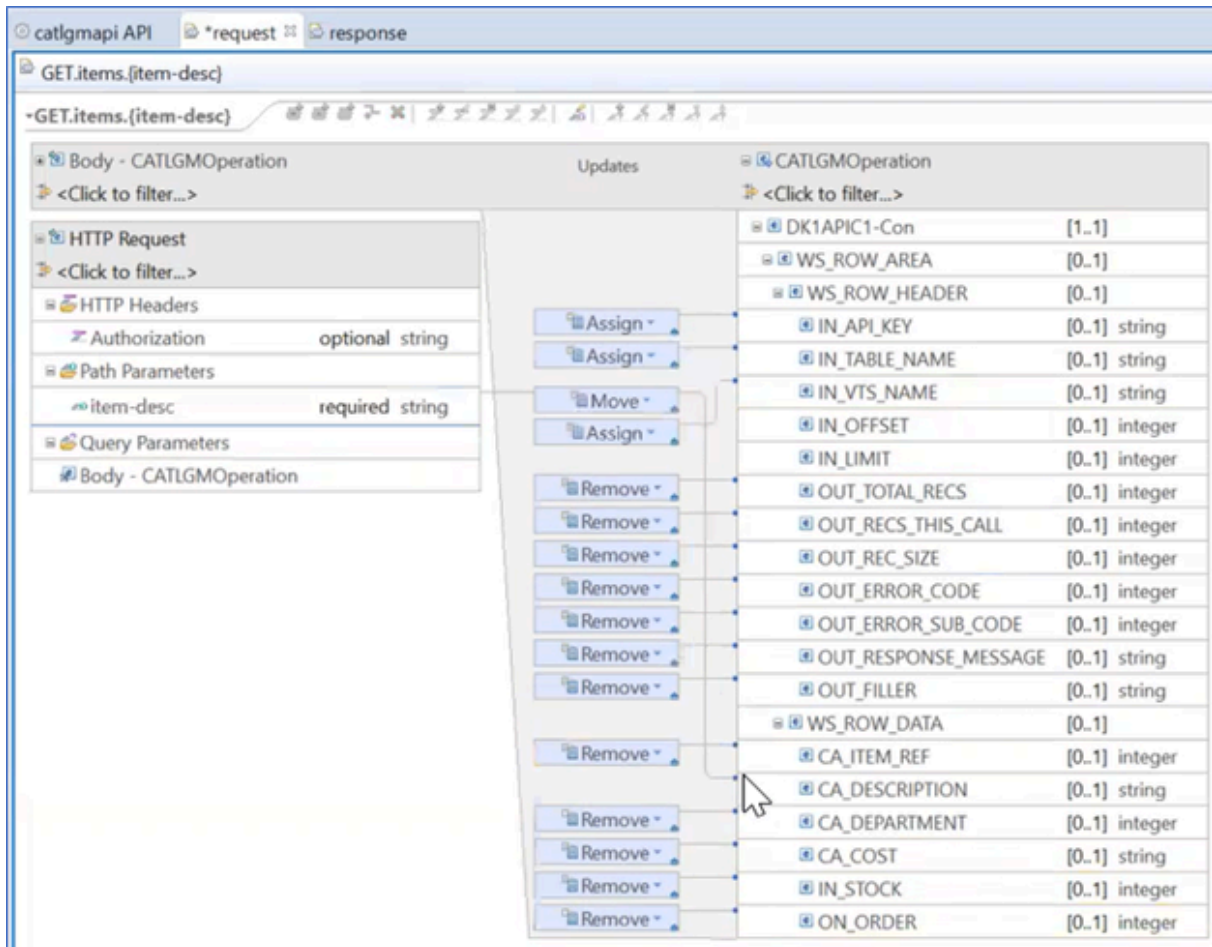
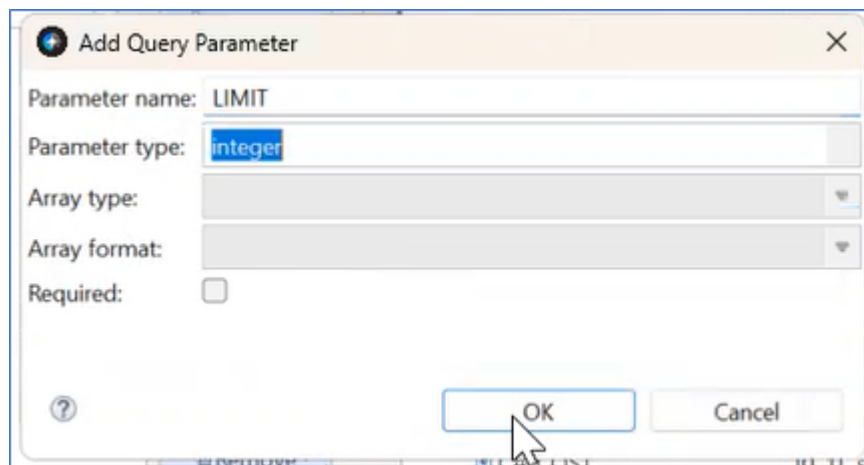


Figure 4-43: Path Parameter item-desc

12. Define the Query Parameters:
 - a. Right-click to select the **Query Parameters** field in the HTTP Request section.
 - b. Select **Add Query Parameter**. The [Add Query Parameter Window](#) will display.
 - c. Enter **LIMIT** in the Parameter Name field.

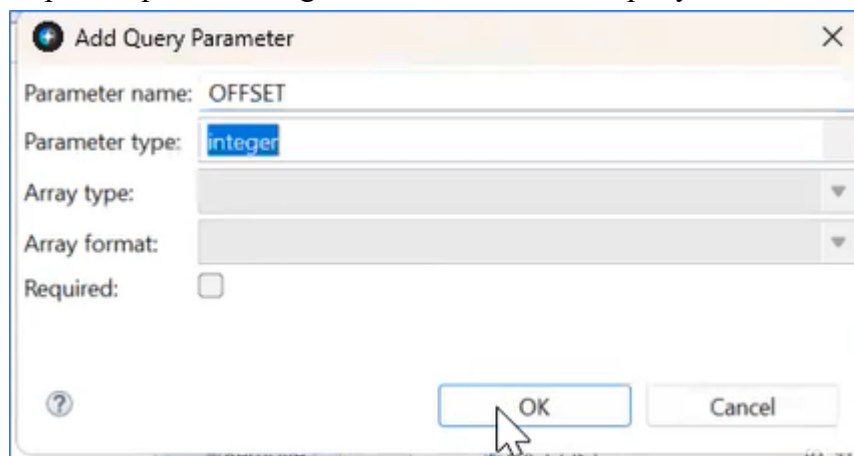
- d. Select **Integer** for the Parameter type.



The screenshot shows a dialog box titled "Add Query Parameter". It has a close button (X) in the top right corner. The "Parameter name" field contains the text "LIMIT". The "Parameter type" dropdown menu is open, and "integer" is selected. Below this, there are two empty dropdown menus for "Array type" and "Array format". A "Required:" checkbox is present and is not checked. At the bottom right, there are "OK" and "Cancel" buttons. A mouse cursor is pointing at the "OK" button.

Figure 4-44: LIMIT Query Parameter

- e. Select the OK button. The [Add Query Parameter window](#) will close.
- f. Repeat steps 12a through 12e for the **OFFSET** query Parameter name.



The screenshot shows a dialog box titled "Add Query Parameter". It has a close button (X) in the top right corner. The "Parameter name" field contains the text "OFFSET". The "Parameter type" dropdown menu is open, and "integer" is selected. Below this, there are two empty dropdown menus for "Array type" and "Array format". A "Required:" checkbox is present and is not checked. At the bottom right, there are "OK" and "Cancel" buttons. A mouse cursor is pointing at the "OK" button.

Figure 4-45: OFFSET Query Parameter

13. Select the [LIMIT Query Parameter](#) and drag it to the **IN_LIMIT** field on the CATLGMOperation.

14. Select the [OFFSET Query Parameter](#) and drag it to the **IN_OFFSET** field on the CATLGMOperation.

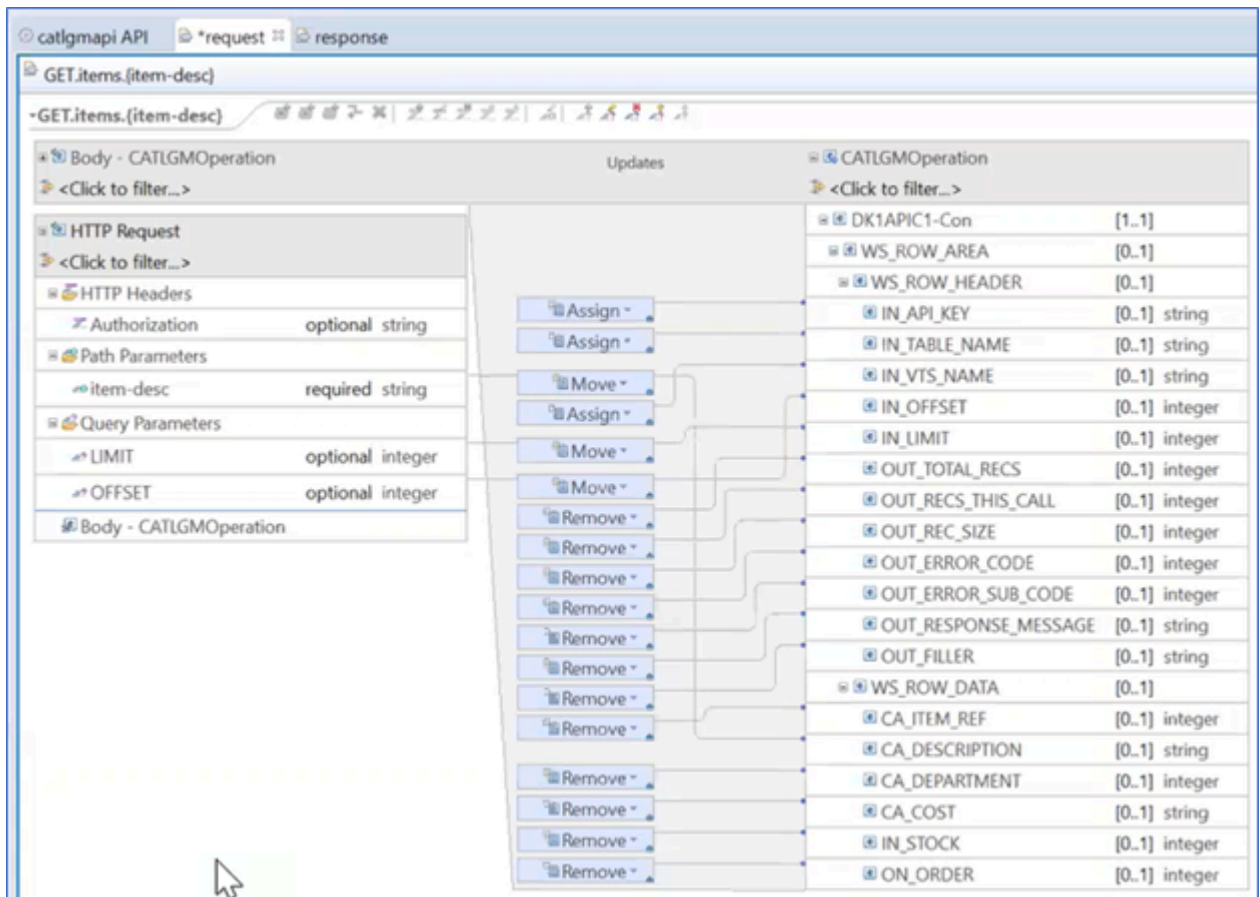


Figure 4-46: Query Parameters assigned

Note: The **IN_TABLE_NAME** and **IN_VTS_NAME** fields can also be supplied as query parameters.

Mapping the GET Response for Multiple Record Retrieval

Mapping the Copybook

1. Expand **DK1APIC1-Con** to display the whole copybook on the right side in the **Body - CATLGMOperationResponse** section.
2. Remove the following fields right-click and select [Add Remove Transform](#):
 - **IN_API_KEY**
 - **IN_TABLE_NAME**
 - **IN_VTS_NAME**
 - **OUT_REC_SIZE**

- OUT_FILLER

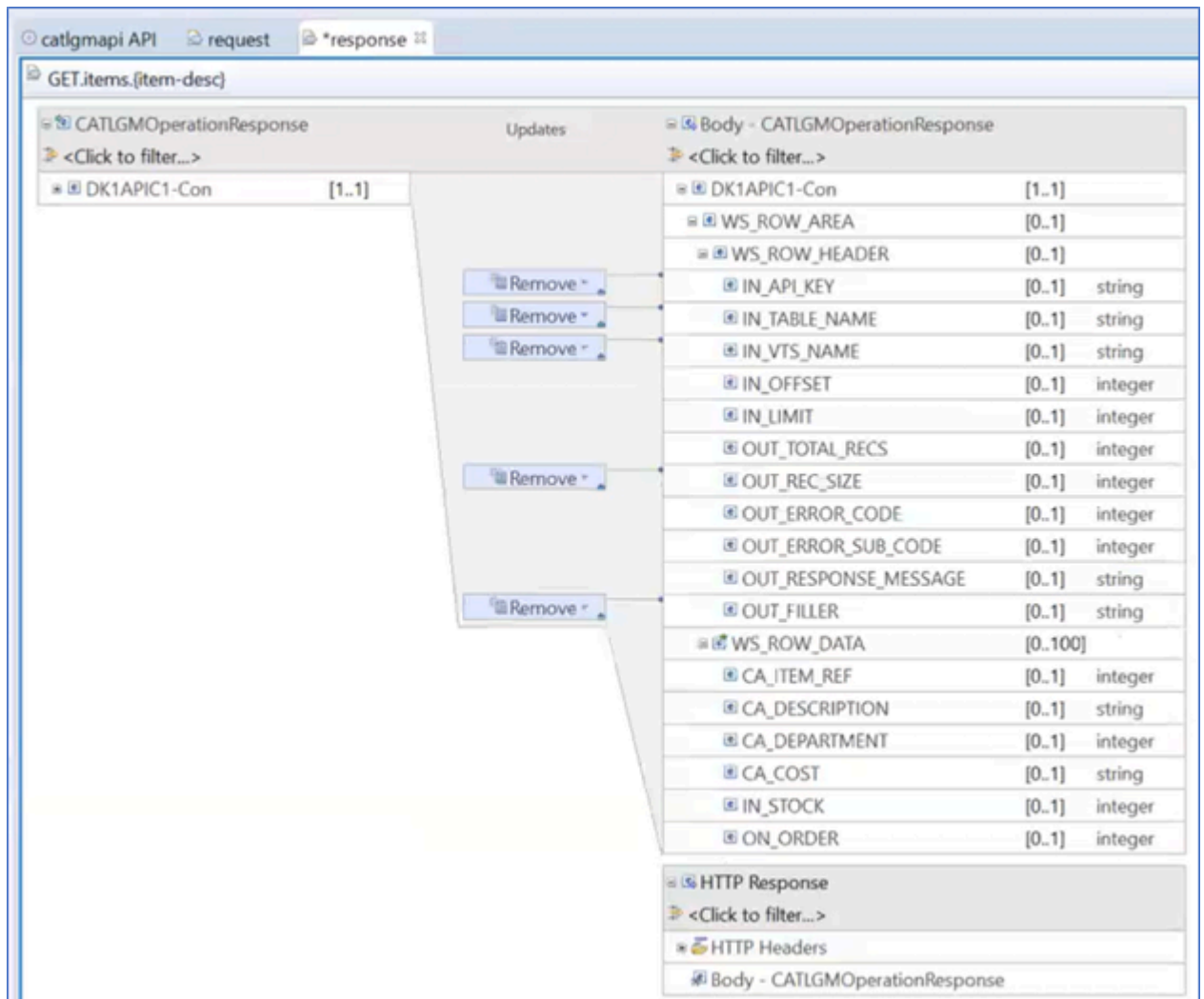


Figure 4-47: Response Mapping the Copybook

Mapping the HTTP Response

The HTTP Header supplies the count of items returned.

1. In the HTTP Response section, right-click to select **HTTP Headers** and select **Add HTTP Header**. The [Add HTTP Header Window](#) will display.

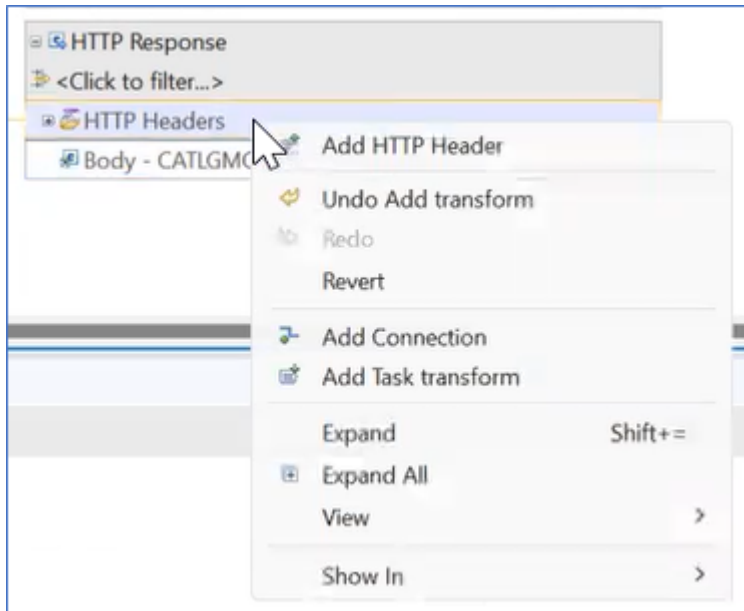


Figure 4-48: Response Add HTTP Header

2. In the Add HTTP Header Window enter:

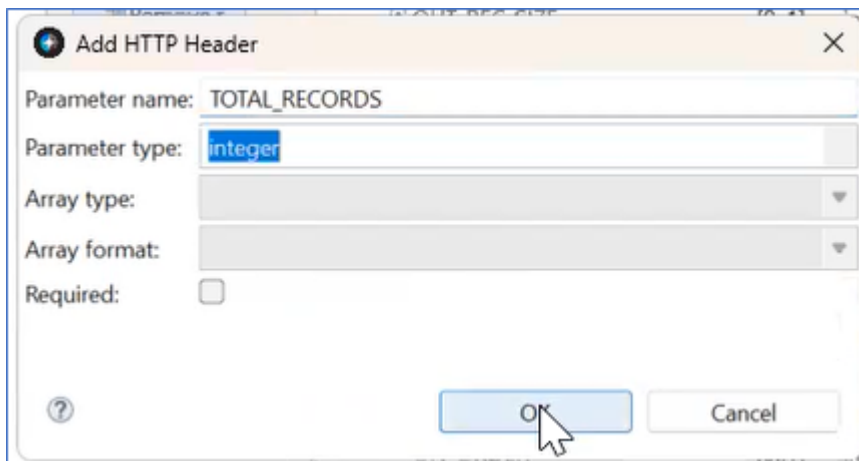


Figure 4-49: Add HTTP Header Window

- a. In the Parameter name enter **TOTAL_RECORDS**.
 - b. In the Parameter type, select **integer**.
 - c. Select the OK button. The [HTTP Header window](#) will close.
3. Expand DK1APIC1-Con to display the whole copybook, on the left side of the response panel, in the **CATLGMOperationResponse** section.

4. Select **OUT_TOTAL_RECS** and drag it to the **TOTAL_RECORDS** HTTP Header. A box showing it is moved and associated to the HTTP Header will display.

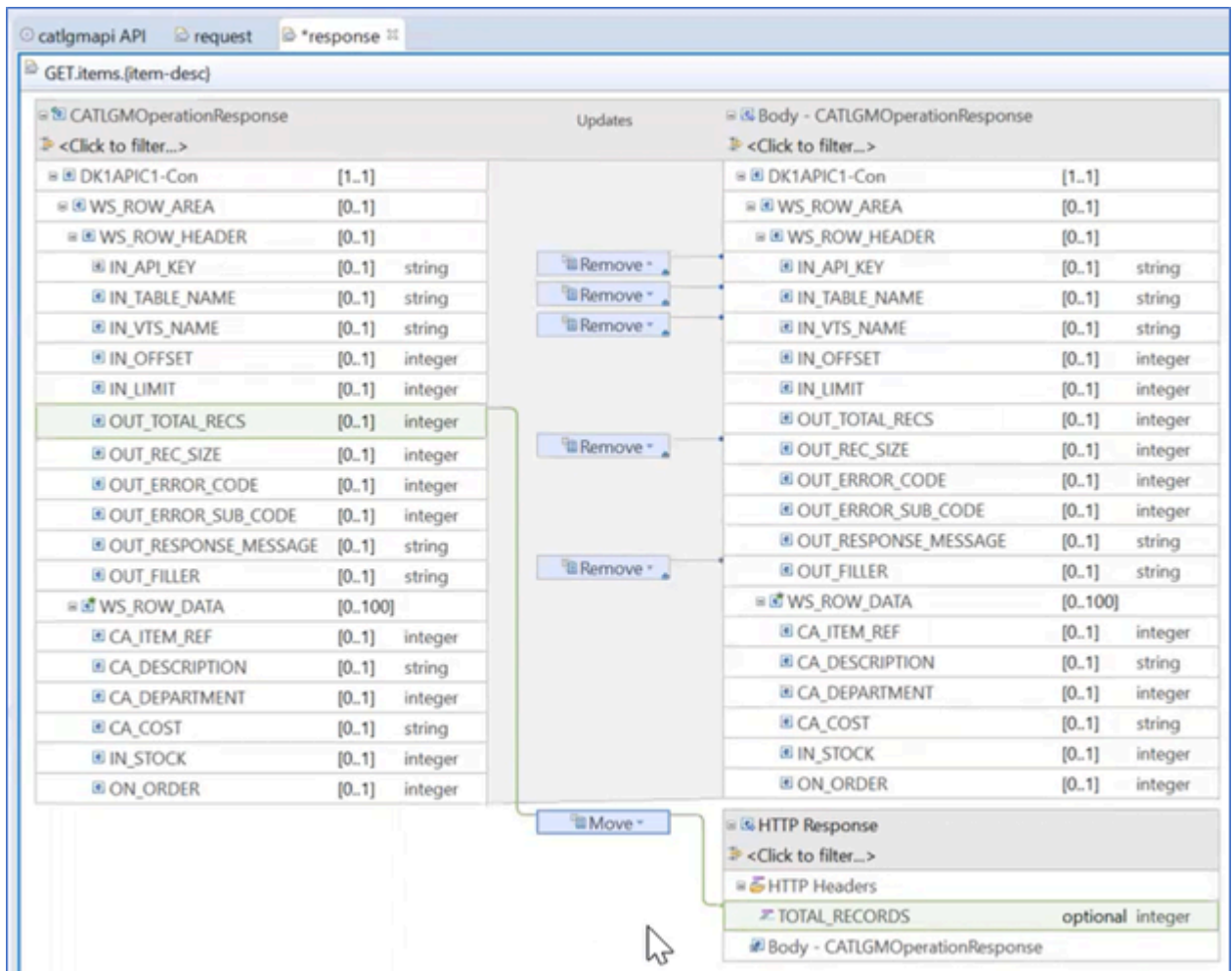


Figure 4-50: OUT_TOTAL_RECS mapped to HTTP Header

Deploy API to server

1. In the Project Explorer panel, right-click to select **CATLGMAPI > z/OS Connect EE > Deploy API to z/OS Connect EE Server**. The **Deploy API window** will display.

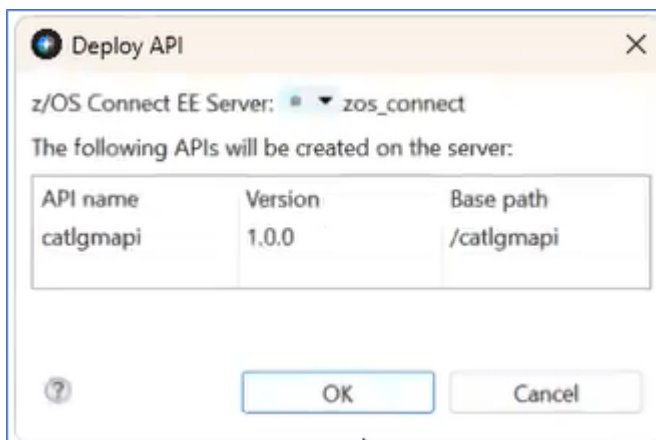


Figure 4-51: Deploy CATLGMAPI

2. Select the OK button. The [Deploy API window](#) will close. The [Deploy API to z/OS Connect EE Server Result](#) window will display.

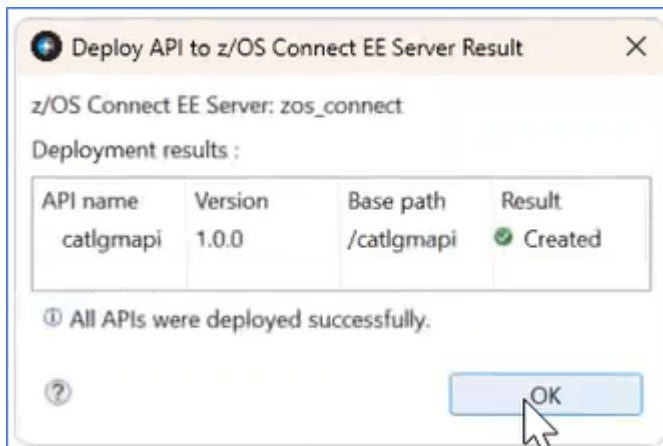


Figure 4-52: Deploy API to zOS Connect EE Server Result

3. Select the OK button. The [Deploy API to z/OS Connect EE Server Result window](#) will close.
4. In the [z/OS Connect EE Servers panel](#) the **catlgmapi** displays.

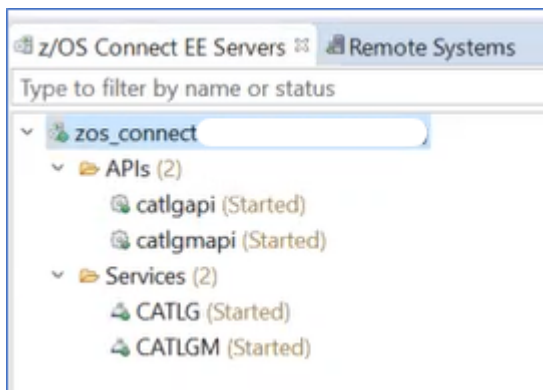


Figure 4-53: catlgmapi started

Create a new Service Project for Multiple Records with Hexadecimal response

Following the instructions described in the section about [Creating A New Service Project for Multiple records](#):

1. Create a new z/OS Connect EE Service Project.
 - a. Provide a Project Name
 - b. Select the CICS Channel Service
 - c. Enter an optional description.
2. On the CICS Configuration enter the Connection Reference to the CICS Region.

Create a New Service Interface for Multiple Records

Following the instructions described for [Create a New Service Request for Multiple Records](#).

1. The Program name is DK1API01
2. Provide a Service Interface Name
3. Edit Container Name to DK1APIC1-Con
4. Import data structure your.prefix.TBDSRC(GENERIC) by browsing the Data structure member and selecting GENERIC.
5. Select GENERIC.si for both the Request and Response Interface fields.

Create New Service Response for Multiple Records with Hexadecimal Response

Follow the instructions for [Create New Service Response for Multiple Records](#).

1. To create the Service Interface enter GENERICM.
2. Edit the Container Name to DK1APIC1-Con
3. Import the data structure your.prefix.CPYCOB(GENERICM) by browsing the Data structure member and selecting GENERICM
4. Disable the OUT_RECS_THIS_CALL
5. Select GENERIC.si for the Request service interface.
6. Select GENERICM.si for the Response service interface.

Deploy The Service Project for Multiple Records with Hexadecimal Response

To Deploy the Service project, follow the instructions for [Deploy Service Project for Multiple Records](#).

New API Project for Multiple Records with Hexadecimal Response

Follow the instructions for [Create a New API Project for Multiple Records](#).

1. Create a new z/OS Connect EE API Project. Provide a name for it.
2. Define the path name.
3. Define the z/OS connect EE Service Archive
4. Remove the POST, PUT, and DELETE methods.
5. Export the Service Package.
6. Select the Workspace Folder.

Mapping the GET Method

The following steps create an operation for multiple record retrieval and multiple record retrieval with a hexadecimal response. For additional information about multiple record retrieval refer to [IBM z/OS Connect](#) topic about how to **Create an operation to inquire about multiple items**.

Map the Service to the GET Method

Using the instructions to Map the [Service to the GET Method](#) instructions earlier in this chapter the following actions will map the service to the GET Method.

1. Select the Service button next to the GET Method
2. Select the workspace to associate the .sar file

Map the Get Request for Multi Responses

Using the instructions earlier in the chapter for how to Map the Get Request, follow these steps:

1. Open both Mappings: request and response.
2. On the Request tab, Expand the DK1APIC1-Con
3. Assign IN_API_KEY to API003
Note: The VTSName and Table Name are query parameters
4. Right-click to select Add Remove Transform for the fields listed in the WS_ROW_HEADER section. Further details can be found earlier in the chapter for [Mapping the Get Request](#).
5. The Path Parameter is mapped to the CA_DATA field
6. The Query Parameters are the Table Name, the VTS Name, Limit and Offset.

Mapping the GET Response for Multiple Record Hexadecimal Retrieval

Mapping the Copybook for Hexadecimal Response

Using the earlier chapter about [Mapping the Copybook](#), follow these instructions.

1. Expand the fields for DK1APIC1-Con.
2. Remove the following fields using the Add Remove Transform function.
 - IN_API_KEY
 - IN_TABLE_NAME
 - IN_VTS_NAME
 - OUT_REC_SIZE
 - OUT_FILLER

Mapping the HTTP Response

Using the [Mapping the HTTP Response](#) instructions earlier in the chapter perform the following actions.

1. Right-click the HTTP Headers and select Add HTTP Header.
2. Set the Parameter name to TOTAL_RECORDS
3. Set the Parameter type to integer.
4. Expand DK1APIC1-Con to display the whole copybook.
5. Select the OUT_TOTAL_RECS and drag it to the TOTAL_RECORDS HTTP Header.

Chapter

5

Testing the RESTConnector

Testing Single Response

The copybooks have been mapped. The **catlgapi** has been deployed. The results of these actions can be tested in a browser using a URL that is **your IP Address:Port/api/explorer**. The browser will display [Liberty REST APIs](#) page.

The new API mapped in z/OS Connect displays.

1. Select the **List Operations** link. The API will display with the GET method displaying the API.



Figure 5-1: REST API GET Method

2. Select the GET method button and the [Response Operation](#) for DK1APIC1-Con will display .

catlgapi

GET /catlgapi/item/{itemsID}

Response Class (Status 200)
OK

Model | Example Value

```
{
  "CATLGOperationResponse": {
    "DK1APIC1-Con": {
      "WS_ROW_AREA": {
        "WS_ROW_HEADER": {
          "OUT_ERROR_CODE": 0,
          "OUT_ERROR_SUB_CODE": 0,
          "OUT_RESPONSE_MESSAGE": "string"
        },
        "WS_ROW_DATA": {
```

Response Content Type

Parameters

Parameter	Value
Authorization	
itemsID	(required)

Figure 5-2: GET button response operation

3. In the itemsID enter the value "20" for one of the items listed in the EXMPCAT table currently loaded in the VTS-TSR for API001.

4. Select the **Try it Out!** button. The mapped copybook will display the results mapped for the ItemsID with the [OUT_RESPONSE_MESSAGE](#) that the operation was successful, and the related data.

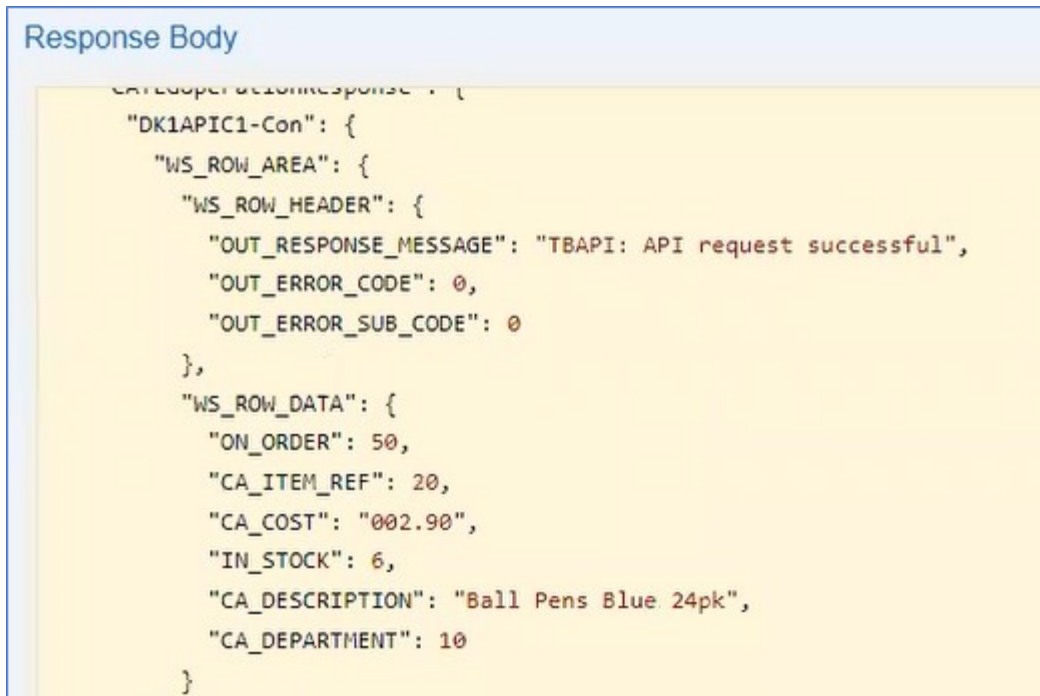


Figure 5-3: Successful Output Response

Testing Multiple Response

1. Select the **List Operations** link. The API will display with the GET method displaying the API.
2. Select the **catlgetapi** GET.
3. In the Parameters section, enter a search option in the **item-desc** field that will generate multiple responses.

4. Select the **Try it Out!** button. In the Response Body the OUT_RESPONSE_MESSAGE displays [API request successful](#). In the Response Headers Section the **total_records = 4**. Four records associated with the item-desc are returned.

Response Body

```
{
  "CATLGMOperationResponse": {
    "DK1APIC1-Con": {
      "WS_ROW_AREA": {
        "WS_ROW_HEADER": {
          "OUT_RESPONSE_MESSAGE": "TBAPI: API request successful",
          "OUT_ERROR_CODE": 0,
          "OUT_ERROR_SUB_CODE": 0,
          "IN_OFFSET": 0,
          "OUT_TOTAL_RECS": 4,
          "IN_LIMIT": 100
        },
        "WS_ROW_DATA": [
          {
            "ON_ORDER": 0,
            "CA_ITEM_REF": 10,
            "CA_COST": "002.90",
            "IN_STOCK": 135,
            "CA_DESCRIPTION": "Ball Pens Black 24pk",
            "CA_DEPARTMENT": 10
          }
        ]
      }
    }
  }
}
```

Response Code

```
200
```

Response Headers

```
{
  "content-language": "en-US",
  "content-length": "736",
  "content-type": "application/json; charset=UTF-8",
  "date": "Fri, 18 Aug 2023 19:09:59 GMT",
  "total_records": "4",
  "x-powered-by": "Servlet/3.1"
}
```

Figure 5-4: Multiple Testing Response Successful

Following these same instructions for the [Multiple Response Retrieval Hexadecimal Data](#). The results will look like this:

```
"IZTAgenericServiceOperationResponse": {
  "DKJAPIC1-Con": {
    "WS_ROW_AREA": {
      "WS_ROW_HEADER": {
        "OUT_RESPONSE_MESSAGE": "DKJAPI: API request successful",
        "OUT_ERROR_CODE": 0,
        "OUT_ERROR_SUB_CODE": 0,
        "OUT_TOTAL_RECS": 43,
        "OUT_REC_SIZE": 32000
      },
      "WS_ROW_DATA": [
        {
          "CA_DATA": "F0F0F0F0F1F00009C3C8D9C9E2E3C9D5C5404040C90004C"
        }
      ]
    }
  }
}
```

Figure 5-5: Multiple Testing for Hexadecimal Response Successful

Appendix

A

Error Messages

This appendix contains all the tableBASE RESTConnector Interface error codes and messages that may be encountered during the normal installation, administration, and operation of the product.

Note: tableBASE RESTConnector internally makes CICS and tableBASE API calls. When an error condition is encountered in either of these internal calls, the API header output fields OUT-ERROR-CODE and OUT-ERROR-SUB-CODE will contain corresponding error messages returned by the API calls.

How to Understand the Error Conditions

The API OUTPUT Field **OUT-RESPONSE-MESSAGE** contains a brief description of the error prefixed with a prefix identifying the type of error.

CICS Errors

CICS errors will display as CICS:Error description/Error encountered during the CICS API Call.

For these CICS Errors the OUT-ERROR-CODE will point to the EIBRESP value. OUT-ERROR-SUB-CODE will point to EIBRESP2 values.

For more information about these CICS errors refer to the linked [CICS Manual on IBM's website](#).

tableBASE Errors

tableBASE errors will display as DK1API Error description during the tableBASE API Call.

For these tableBASE Errors the OUT-ERROR-CODE will point to tableBASE COMMAND-AREA filed ERROR. The TB-ERROR-CODE will point to TB-ERROR-SUB-CODE.

Refer to **tableBASE Programming Guide** Appendix E for a detailed explanation of these error codes and their meaning.

API Errors

API errors will display as DK1API:Error description by API Interface module.

Refer to table [Table A-1](#) on page 80 provided in this manual.

Note: All API Interface generated error codes have negative values to differentiate them from tableBASE error codes.

Error Codes

OUT-ERROR-CODE / OUT-ERROR-SUB-CODE	Text	Meaning / Instructions
-01/00	Input Size Error – Size exceeds the limit	API Interface can handle the record size up to 32K. Size exceeds the limit. Check the size of the COPYBOOK that is customized for this API.
-02/00	Input Size Error – Size of the API Specific area is less than or equal to Zero.	COPYBOOK should contain HEADER AREA and API Specific area. Check the size of the COPYBOOK that is customized for this API.
-03/00	Input COPYBOOK Header field IN-LIMIT has invalid value.	API can only retrieve 100 records at a time. Check the value passed to IN-LIMIT
-04/00	API not found in APICONFG table.	Check the value passed in for COPYBOOK Header field IN-API-KEY
-05/00	Invalid API Type found in APICONFG table.	Check the value for API-TYPE field in APICONFG table for the COPYBOOK Header field IN-API-KEY value passed in.
-06/00	Table Not found in tableBASE	Table name passed in COPYBOOK header field IN-TABLE-NAME was not found.
-07/00	Item Not found – Single record retrieval	HTTP GET operation could not find the item in the tableBASE table.
-08/00	Item Not found – Multiple record retrieval	HTTP GET operation could not find the Item matching the Query parameter to retrieve multiple records.

OUT-ERROR-CODE / OUT-ERROR-SUB-CODE	Text	Meaning / Instructions
-09/00	Unknown tableBASE command found in APICONFG table	Check the value in APICONFG table for the IN- API-KEY value passed into the API call.
-10/00	Record Size exceeded.	Maximum allowed size for Hexadecimal table row size is 16000 bytes. This is only applicable to API003

Table A-1: Error Messages

