

Leveraging Web Services – A Real World Example

A vibrant, abstract banner for the Adobe ColdFusion Summit 2016. The background is a complex, multi-colored geometric pattern of triangles and polygons in shades of green, blue, and yellow. A large, stylized 'CF' logo is prominently displayed on the right side, rendered in a 3D, multi-colored font. The overall aesthetic is modern and tech-oriented.

Adobe ColdFusion
Summit 2016

October 10-11, Las Vegas



ThermoFisher
S C I E N T I F I C

Keen Haynes

Senior ColdFusion Engineer
17+ years with ColdFusion

keen.haynes@thermofisher.com

Doug Rehg

Senior ColdFusion Engineer
15+ years with ColdFusion

doug.rehg@thermofisher.com

This is not a “best practices” discussion but real world overview of how we did it at ThermoFisher and how we changed our approach with CF2016 and the API Manager

Agenda

- Soap Example
- RESTful Example
- Our Challenge
- Decisions We Faced
 - REST VS SOAP
 - Authentication
 - Tracking
- Leveraging CF 2016 and the API Manager
 - Security
 - Testing
 - Swagger Docs
 - Dashboard (Tracking)
- Application Demo
- Questions

Basic SOAP

```
<cfcomponent output="false" style="document">
  <cffunction name=" getAuthorDetails " access="remote" output="no" returntype=" query"
  hint="returns author details">

    <cfargument name="authorid" required="false" default="0" type="numeric">

    <cfquery name="qAuthorDetails" datasource="#request.readDSN#">
      select * from app.authors
      <cfif arguments.authorid>
        where authorid = #arguments.authorid#
      </cfif>
    </cfquery>
    <cfreturn qAuthorDetails>
  </cffunction>
</cfcomponent>
```

Calling SOAP

Pass Optional argument with value

```
<cfinvoke webservice="http://localhost/webservices/authorSoap.cfc?wsdl"
  method=" getAuthorDetails " returnvariable="authDetails">
  <cfinvokeargument name="authorid" value="5" />
</cfinvoke>
```

Pass Optional argument without value

```
<cfinvoke webservice="http://localhost/webservices/authorSoap.cfc?wsdl"
  method=" getAuthorDetails " returnvariable="authDetails">
  <cfinvokeargument name="authorid" value="" omit="true" />
</cfinvoke>
```

Basic REST

```
<cfcomponent rest="true" restpath="/authors" displayname="authors" >  
  
    <cffunction name="getAuthorDetails" access="remote" httpmethod="GET"  
    returntype="any" restpath="{authorid}" produces="application/xml">  
  
        <cfargument name="authorid" required="true" restargsource="Path"/>  
        <cfquery name="qAuthorDetails" datasource="#request.readDSN#">  
            select * from app.authors  
            where authorid = #arguments.authorid#  
        </cfquery>  
        <cfreturn qAuthorDetails>  
    </cffunction>  
</cfcomponent>
```

Register Your REST Service - ColdFusion 10

The screenshot shows the Adobe ColdFusion 10 Administrator web interface. The browser address bar indicates the URL is `http://localhost:8500/authorID.cfm`. The page title is "Data & Services > REST Services".

SERVER SETTINGS

- Settings
- Request Tuning
- Caching
- Client Variables
- Memory Variables
- Mappings
- Mail
- Scheduled Tasks
- WebSocket
- Chatting
- Font Management
- Document
- Java and JVM
- Settings Summary

DATA & SERVICES

- Data Sources
- ColdFusion Collections
- Solr Server
- Web Services
- REST Services** (indicated by a red arrow)
- Flex Integration

DEBUGGING & LOGGING

- SERVER MONITORING
- EXTENSIONS
- EVENT GATEWAYS
- SECURITY
- PACKAGING & DEPLOYMENT
- ENTERPRISE MANAGER
- SERVER UPDATE

Data & Services > REST Services

Register your applications and folders to generate RESTful web services. ColdFusion automatically registers CFCs found in the registered folders.

Add/Edit REST Service

Root path
Application path or root folder where ColdFusion searches for CFCs

Service Mapping
Alternate string to be used for application name while calling REST service. For example, `http://localhost/rest/(service mapping)/test` (Optional)

Set as default application
There can be only one default application (which can be changed later). Setting an application as default makes application name optional while calling the webservice. For example `http://localhost/rest/path`

Active ColdFusion REST Services

Actions	Root Path	Service Mapping	Default
<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	C:\Server\Apache2.2\htdocs\summitRest	summitRest	

© 1997 - 2012 Adobe Systems Incorporated and its licensors. All Rights Reserved.
Notices, terms and conditions pertaining to third party software are located at <http://www.adobe.com/go/thirdparty> and incorporated by reference herein.

Register Your REST Service - CF2016

Server: cfusion
Expand All / Collapse All

SERVER SETTINGS

- Settings
- Request Tuning
- Caching
- Client Variables
- Memory Variables
- Mappings
- Mail
- Scheduled Tasks
- WebSocket
- Charting
- Font Management
- Document
- Java and JVM
- Settings Summary

DATA & SERVICES

- Data Sources
- ColdFusion Collections
- Solr Server
- Web Services
- REST Services**
- Flex Integration
- PDF Service

DEBUGGING & LOGGING

- SERVER MONITORING
- EXTENSIONS
- EVENT GATEWAYS
- SECURITY
- PACKAGING & DEPLOYMENT
- ENTERPRISE MANAGER
- SERVER UPDATE

Server has been updated successfully

Data & Services > REST Services

Register your applications and folders. ColdFusion automatically registers CFCs found in the registered folders.

Add/Edit REST Service

Root path
Application path or root folder where CFCs reside

Host ←
Host name for the REST service. Example: localhost:8500 (Optional)

Service Mapping
Alternate string to be used for application name while calling REST service. Example: http://localhost/rest/{service mapping}/test (Optional)

Set as default application
Set an application as default to exclude the application name in the URL while calling the web service. One default application is allowed for a host.
Example http://localhost/rest/path

Active ColdFusion REST Services

Actions	Root Path	Service Mapping	Default	Host
<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	C:/ColdFusion2016/cfusion/wwwroot/rest2016	rest2016	NO	localhost:8500

© 1997 - 2016 Adobe Systems Incorporated and its licensors. All Rights Reserved.
Notices, terms and conditions pertaining to third party software are located at <http://www.adobe.com/go/thirdparty/> and incorporated by reference herein.

Calling Your REST Service

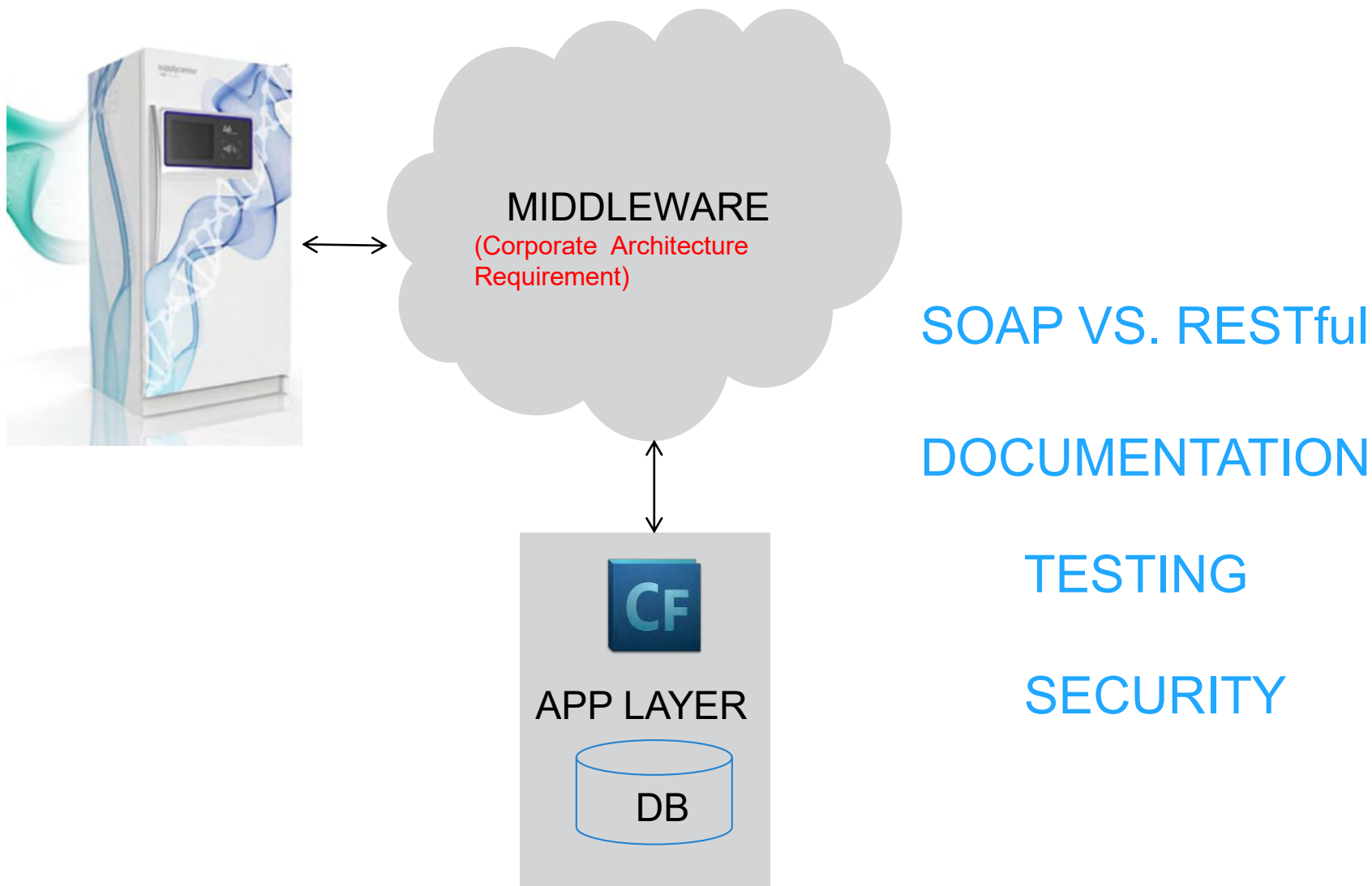
- `<cfhttp url="http://localhost/rest/summitRest/authors/1" result="restResult" method="get">`
- `</cfhttp>`

Our Challenge



- Provide our customers access to Thermo Fisher products real time on their own campus
- Make it super simple for customers to purchase and retrieve our products
- The products needed to be securely stored and refrigerated
- Inventory and replenishment needed to be automatic

Decisions We Faced



Soap

- RESTful not supported by CF8
- Middleware utilized SOAP

RESTful

- CF 10 introduced REST support
- Approval to bypass middleware

Initial Solution – SOAP VS. RESTful

SOAP

Documentation

- WSDL

Testing

- Custom CFML Test Harnesses
- SOAP UI

Security

- Security handled by Oracle OFM Middleware
– Server Recognition

RESTful

Documentation

- No self documenting tools for REST
- Unfamiliar with SWAGGER
- Hand coded in CFML

Testing

- Custom CFML Test Harnesses
- POSTMAN / BOOMERANG

Security

- Coded

Initial Solution – Books RESTful Code Snippet

```
<cfcomponent rest="true" restpath="/books" displayname="books" extends="packetTracking">
  <cffunction name="getBookDetailsJSON" access="remote" httpmethod="GET" returntype="any"
    restpath="{bookid}" produces="application/json">
    <cfargument name="bookid" required="true" restargsource="Path" type="string">
    <cfinclude template="inc_tracking.cfm">
    <cfquery name="qBookDetails" datasource="#request.readDSN#">
      select b.bookid, a.firstname, a.lastname, a.authorid, b.title, b.bookimage, b.bookdescription,
        b.isspotlight, b.genre
      from app.books b, app.authors a
      where a.authorid = b.authorid
      and b.bookid = #arguments.bookid#
    </cfquery>
    <cfset temp = restThrow(throw="false", errorCode="200", message="OK", detail="Action completed
      successfully.", data="#serializeJSON(qBookDetails)#", sourcename="#getfunctioncalledname()#")>
    <cfreturn qBookDetails>
  </cffunction>
```

Initial Solution – inc_tracking.cfm Snippet

```
<cfset httpRequest = getHTTPRequestData()>
<cfset request.rest = {}>
<cfset request.rest["sessionId"] = createUUID()>
<cfset request.rest["tsStart"] = now()>
<cfset request.rest["restPath"] = CGI.PATH_INFO>

<cfif isDefined('httpRequest.Headers.Authorization')>
  <cfset request.rest["authString"] = ToString(ToBinary(
ListLast(GetHTTPRequestData().Headers.Authorization, " ") ))>
  <cfset request.rest["memberEmail"] = ListFirst(request.rest.authString,":")>
  <cfset request.rest["memberPassword"] = ListLast(request.rest.authString,":")>
<cfelse>
  <cfset request.rest["authString"] = "NA">
  <cfset request.rest["memberEmail"] = "NA">
  <cfset request.rest["memberPassword"] = "NA">
</cfif>

<cfset theDir=GetDirectoryFromPath(GetCurrentTemplatePath())>
<cfset packetFileName = theDir & "packetTracker.xls">
```


Initial Solution – inc_tracking.cfm cont.

```
<cfif FileExists(packetFileName)>
```

```
  <cfspreadsheet action="read" src="#packetFileName#" sheetname="packets"  
    name="packetTracker">
```

```
  <cfset spreadsheetAddRow(packetTracker,"#request.rest.sessionId#,#request.rest.restPath# ,  
    #request.rest.authString#,,,,#request.rest.tsStart#",2,1)>
```

```
  <cfspreadsheet action="write" filename="#packetFileName#" name="packetTracker"  
    sheetname="packets" overwrite=true>
```

```
</cfif>
```

```
  <cfset theSheet = SpreadsheetNew("packetTracker")>
```

```
  <cfset spreadsheetAddRow(theSheet,"sessionId,restPath,authString,responseStatusCode,  
    responseStatusMessage,responseStatusDetail,responseData,tsStart,tsEnd",1,1)>
```

```
  <cfset spreadsheetAddRow(theSheet,"#request.rest.sessionId#,#request.rest.restPath#,  
    #request.rest.authString#,,,,#request.rest.tsStart#",2,1)>
```

```
  <cfspreadsheet action="write" filename="#packetFileName#" name="theSheet"  
    sheetname="packets" overwrite=true>
```

```
</cfif>
```

Initial Solution – tracking.cfc Snippet

```
<cffunction name="restThrow" returntype="STRING">
```

```
<cfargument name="errorCode" type="any" default="" required="false">
```

```
<cfargument name="message" type="string" default="" required="false">
```

```
<cfargument name="detail" type="string" default="" required="false">
```

```
<cfargument name="data" type="string" default="" required="false">
```

```
<cfargument name="throw" type="boolean" default="true" required="false" hint="determines  
whether or not to throw the actual exception">
```

```
<cfargument name="sendNotification" type="string" default="AllNon200s">
```

```
<cfargument name="sourcename" type="string" default="none">
```

```
<cfset returnval = serializeJSON(arguments)>
```

```
<cfset theDir=GetDirectoryFromPath(GetCurrentTemplatePath())>
```

```
<cfset packetFileName = theDir & "packetTracker.xls">
```

```
<cfspreadsheet action="read" src="#packetFileName#" sheetname="packets"  
name="theSheet">
```

```
<cfset spreadsheetAddRow(theSheet, #request.rest.sessionId#, #arguments.errorCode#,  
#arguments.message#, #arguments.detail#, #arguments.data#, #NOW()#, 2, 1)>
```

```
<cfspreadsheet action="write" filename="#packetFileName#" name="theSheet"  
sheetname="packets" overwrite=true>
```


Initial Solution – Books RESTful Code Snippet

```
<cffunction name="addBook" access="remote" httpmethod="POST" returntype="any"
restpath="bookAdd">
    <cfargument name="structData" type="struct" required="true" >

    <cfinclude template="inc_tracking.cfm">
    <cfinclude template="inc_auth.cfm"> ← Security Call

    <cfquery name="qAddBook" datasource="#request.readDSN#" result="insertResults">
        insert into app.books (authorid, title, bookimage,thumbnailimage,
bookdescription, isspotlight, genre)
        values (#structData.authorid#, '#trim(structData.title)#',
'#trim(structData.bookimage)#', '#trim(structData.thumbnailimage)#',
'#trim(structData.bookdescription)#', '#structData.isspotlight#',
'#trim(structData.genre)#')
    </cfquery>

    <cfset temp = restThrow(throw="false", errorCode="200", message="OK", detail="Action
completed successfully.", data="#serializeJSON(insertResults)#",
sourcename="#getfunctioncalledname()#")>

    <cfreturn insertResults>
</cffunction>
```

Initial Solution – Calling API With Security

```
<cfset variables.startStr = "super@demodata.com:demo">
<cfset variables.base64Str = toBase64(variables.startStr)>

<cfset variables.jsonStr = {firstname="John", lastname="Stone", bio="John is from Carlsbad CA",
isspotlight = "Y"}>
<cfset variables.dataString = SerializeJSON(variables.jsonStr)>
<cfhttp url="http://localhost/rest/summitRest/authors/authorAdd" result="restResult" method="post">
  <cfhttpparam TYPE="Header"
    name="content-type"
    value="application/JSON">
    <cfhttpparam type="Header"
      name="Authorization"
      value="#variables.base64Str#">
  <cfhttpparam TYPE="Body"
    name="structData"
    value="#variables.dataString#">

</cfhttp>
<cfdump var="#restResult#">
```

Initial Solution – inc_auth.cfm

```
<cfset LOCAL.EncodedCredentials = GetHTTPRequestData().Headers.Authorization>  
<cfset LOCAL.Credentials = ToString(ToBinary( LOCAL.EncodedCredentials ))>  
<cfset LOCAL.memberEmail = ListFirst( LOCAL.Credentials, ":" )>  
<cfset LOCAL.memberPassword = ListLast( LOCAL.Credentials, ":" )>
```

```
<cfquery name="getMember" datasource="#request.readDSN#">
```

```
  select *
```

```
  from app.members
```

```
  where lower(email)='#lcase(LOCAL.memberEmail)##'
```

```
  and memberpassword='#LOCAL.memberPassword##'
```

```
  and isactive = 'Y'
```

```
</cfquery>
```

```
<cfif getMember.recordcount EQ "0">
```

```
  <cfthrow errorcode="401" message="Unauthorized!" type="401" detail="Username or  
Password incorrect">
```

```
<cfelse>
```

```
  <cfset arguments.memberid = getMember.memberid>
```

```
  <cfset arguments.memberfname = getMember.firstname>
```

```
  <cfset arguments.memberlname = getMember.lastname>
```

```
  <cfset arguments.memberemail = getMember.email>
```

```
  <cfset arguments.memberisadmin = getMember.isadmin>
```

```
</cfif>
```

Then Came CF 2016 And The API Manager

Create API – Registered REST services

The screenshot shows the Adobe ColdFusion 2016 Release Administrator interface. The main content area is titled "Data & Services > REST Services". It contains a section for "Add/Edit REST Service" with fields for "Root path", "Host", and "Service Mapping". Below this is a section for "Active ColdFusion REST Services" which contains a table with columns for "Actions", "Root Path", "Service Mapping", "Default", and "Host". The table lists three services: "restTest3", "restTest", and "restMobile". The "restTest3" and "restMobile" rows are highlighted with a red border.

Server: cfusion
Expand All / Collapse All

SERVER SETTINGS

- Settings
- Request Tuning
- Caching
- Client Variables
- Memory Variables
- Mappings
- Mail
- Scheduled Tasks
- WebSocket
- Charting
- Font Management
- Document
- Java and JVM
- Settings Summary

DATA & SERVICES

- Data Sources
- ColdFusion Collections
- Solr Server
- Web Services
- REST Services
- Flex Integration
- PDF Service

DEBUGGING & LOGGING

SERVER MONITORING

EXTENSIONS

EVENT GATEWAYS

SECURITY

PACKAGING & DEPLOYMENT

ENTERPRISE MANAGER

SERVER UPDATE

Data & Services > REST Services

Register your applications and folders. ColdFusion automatically registers CFCs found in the registered folders.

Add/Edit REST Service

Root path
Application path or root folder where CFCs reside

Host
Host name for the REST service. Example: localhost:8500 (Optional)

Service Mapping
Alternate string to be used for application name while calling REST service. Example: http://localhost/rest/(service mapping)/test (Optional)

Set as default application
Set an application as default to exclude the application name in the URL while calling the web service. One default application is allowed for a host.
Example http://localhost/rest/path

Active ColdFusion REST Services

Actions	Root Path	Service Mapping	Default	Host
<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	D:\web\commerce.qa2.invitrogen.com\scms\rest3	restTest3	NO	
<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	D:\web\commerce.qa2.invitrogen.com\scms\rest2	restTest	YES	
<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	D:\web\commerce.qa2.invitrogen.com\scms\restMobile	restMobile	NO	

© 1997 - 2016 Adobe Systems Incorporated and its licensors. All Rights Reserved.
Notices, terms and conditions pertaining to third party software are located at <http://www.adobe.com/go/thirdparty/> and incorporated by reference herein.

CF2016 Administrator – Allow REST Discovery

The screenshot shows the Adobe ColdFusion 2016 Release Administrator web interface. The browser address bar shows the URL `127.0.0.1:8500/CFIDE/administrator/index.cfm`. The left sidebar contains a navigation menu with categories: SERVER SETTINGS, DATA & SERVICES, DEBUGGING & LOGGING, SERVER MONITORING, EXTENSIONS, EVENT GATEWAYS, SECURITY, PACKAGING & DEPLOYMENT, ENTERPRISE MANAGER, and SERVER UPDATE. The 'SERVER SETTINGS' category is expanded, and the 'Settings' link is highlighted with a red arrow. The main content area displays various configuration sections: 'Allowed file extensions for CFInclude tag' (with a text input field containing '*'), 'Application.cfc/Application.cfm lookup order' (with radio buttons for 'Default order', 'Until webroot', and 'In webroot'), 'Error Handlers' (with sub-sections for 'Missing Template Handler' and 'Site-wide Error Handler'), and 'Request Size Limits' (with input fields for 'Maximum number of POST request parameters' set to 100, 'Maximum size of post data' set to 20 MB, 'Request Throttle Threshold' set to 4 MB, and 'Request Throttle Memory' set to 200 MB). The 'API Manager' section at the bottom has the checkbox 'Allow REST Discovery' checked, which is also highlighted with a red arrow. Below this checkbox is the text: 'Specify whether to allow API Manager to discover REST services published in ColdFusion.' At the bottom right of the main content area is a 'Submit Changes' button. The footer of the page contains the copyright notice: '© 1997 - 2016 Adobe Systems Incorporated and its licensors. All Rights Reserved.'

API Manager Administrator – Server Discovery

Note: port number by default is 9000

API Manager Administrator

Welcome admin Logout

Server

Security

SLA

Cluster

Caching

CF Discovery Server

Notifications

Log

Analytics Dashboard

CF Discovery Server Configuration

Add New Server

Protocol: HTTP

Name*

Address*
Host: Port:

Context*

Username*

Password*

Reset Add Server

Existing Servers

Name	Protocol	Host	Port	Context	Username	Edit	Delete
DEV2046	HTTP	localhost	8500	rest	admin		
usfrd-scapp1-d4	HTTP	usfrd-scapp1-d4.devinvitrogen.net	80	rest	admin		

Create API - Import REST API From ColdFusion

The screenshot displays the ColdFusion API Manager Portal interface. The browser address bar shows the URL `localhost:9000/portalU#/create`. A red arrow points to the port number `9000` in the address bar, with a red text annotation: **Note: port number by default is 9000**. The page header includes the ColdFusion logo and the text "API Manager Portal" on the left, and the user name "admin" with a help icon on the right. A left-hand navigation menu contains the following items: "My APIs", "Create API", "API Catalog", "Subscriptions", "Notifications", "Key Stores", and "Analytics". A red arrow points to the "Create API" menu item. The main content area features four large, light-blue buttons with circular icons: "Create REST API" (plus icon), "Import REST API From ColdFusion" (download icon with "CF" in a circle, highlighted with a red box), "Import REST API From Swagger" (download icon with "S" in a circle), "Create REST API From SOAP" (plus icon with "SOAP" in a circle), and "Import SOAP API" (download icon with "SOAP" in a circle).

Create API - Select RESTful Service to Import

The screenshot displays the API Manager Portal interface. A modal dialog titled "Import REST API From ColdFusion" is open, showing a tree view of services. The tree is expanded to show two categories: "usfrd-scapp1-d4.dev.invitrogen.net:80" and "localhost:8500". Under "usfrd-scapp1-d4.dev.invitrogen.net:80", there are three services: "restTest3", "DefaultApplication", and "restMobile". Under "localhost:8500", there is one service: "rest2016". Each service has a yellow arrow icon and an "Import" button. The dialog also has "Cancel", "Save", and "Publish" buttons at the top right. In the background, the "Basic" configuration page is visible, showing a "Logo" field with an "API" icon, a "Description" field, and a "Browse..." button. A tooltip labeled "External Import" is visible on the right side of the dialog.

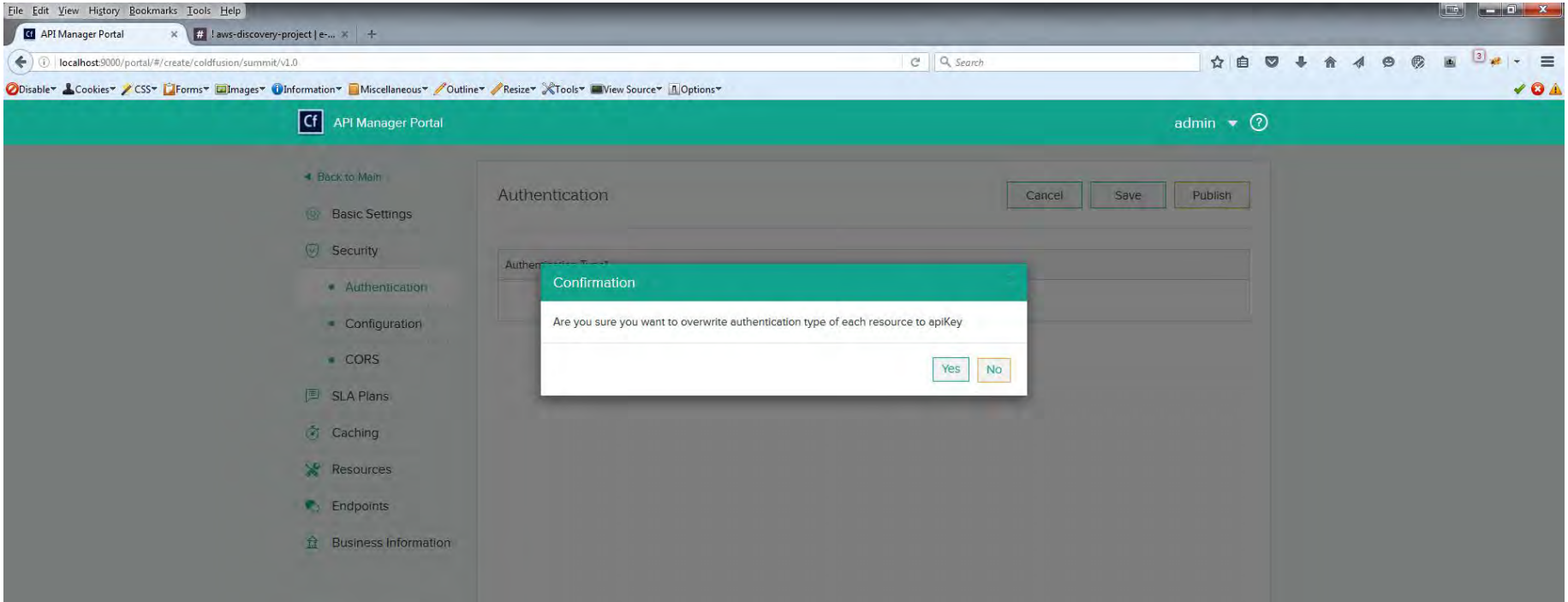
Create API - REST Import Complete

The screenshot shows the 'API Manager Portal' interface. The top navigation bar includes the 'API Manager Portal' logo and the user 'admin'. A left sidebar contains a menu with options: 'Back to Main', 'Basic Settings', 'Security' (with sub-items 'Authentication', 'Configuration', 'CORS'), 'SLA Plans', 'Caching', 'Resources', 'Endpoints', and 'Business Information'. The main content area displays the 'Basic Settings' form for an API named 'rest2016'. The form fields are: 'API Name' (rest2016), 'Context' (rest2016), 'Version' (v1.0) with a 'Make Default' checkbox, 'Visibility' (Public), 'Logo' (API icon) with 'Browse...' and 'Upload' buttons, and 'Description' (Enter description). Action buttons 'Cancel', 'Save', and 'Publish' are at the top right of the form. A green notification box in the bottom right corner states: 'Resources have been generated which can be found in resources section'.

Create API - Apply Authentication

The screenshot shows the 'Authentication' configuration page in the API Manager Portal. The page title is 'Authentication' and it includes 'Cancel', 'Save', and 'Publish' buttons. The 'Authentication Type*' section contains a 'Type*' dropdown menu with the following options: 'apiKey', 'oAuth2', 'basicAuth', 'none', and 'apiKey'. The 'Apply to all resources' checkbox is checked. The 'Authentication' menu item in the left sidebar is highlighted with a red box. A red arrow points to the 'apiKey' option in the dropdown menu.

Create API - Confirm Authentication



Create API – Select Resources

The screenshot displays the API Manager Portal interface. The top navigation bar includes the logo and 'API Manager Portal' on the left, and 'admin' with a help icon on the right. A left-hand sidebar contains a list of menu items: 'Back to Main', 'Basic Settings', 'Security', 'Authentication', 'Configuration', 'CORS', 'SLA Plans', 'Caching', 'Resources' (highlighted with a red box), 'Endpoints', and 'Business Information'. The main content area is titled 'Authentication' and features 'Cancel', 'Save', and 'Publish' buttons. Below the title, there is a section for 'Authentication Type*' with a dropdown menu currently set to 'apiKey' and an 'Apply to all resources' button. A green notification box in the bottom right corner contains a checkmark and the text 'Authentication has been changed for each sub resource'.

Create API – Select Get

The screenshot displays the API Manager Portal interface. On the left, a navigation menu includes 'Back to Main', 'Basic Settings', 'Security', 'Configuration', 'CORS', 'SLA Plans', 'Caching', 'Resources', 'Endpoints', and 'Business Information'. The 'Resources' section is active, showing three resource configurations:

- /books2016** (Mapped Name: /books2016):
 - POST: /books2016/bookAdd
 - GET: /books2016/{bookid} (highlighted with a red arrow)
 - DELETE: /books2016/{bookid}
 - PUT: /books2016/bookUpdate
 - GET: /books2016/bookList
- /authors2016A** (Mapped Name: /authors2016A):
 - GET: /authors2016A/{authorid}
 - DELETE: /authors2016A/{authorid}
 - PUT: /authors2016A/authorUpdate
 - POST: /authors2016A/authorAdd
 - GET: /authors2016A/authorsList
- /authors2016** (Mapped Name: /authors2016):
 - GET: /authors2016/{authorid}
 - DELETE: /authors2016/{authorid}
 - PUT: /authors2016/authorUpdate
 - POST: /authors2016/authorAdd

Each resource configuration includes 'Add Operation' and 'Manage Models' buttons. The top right of the main content area has 'Cancel', 'Save', and 'Publish' buttons. The browser's address bar shows 'localhost:9000/portal/#/create/coldfusion'.

Set Authentication To "None"

The screenshot shows the API Manager Portal interface. On the left is a navigation menu with options like 'Back to Main', 'Basic Settings', 'Security', 'Authentication', 'Configuration', 'CORS', 'SLA Plans', 'Caching', 'Resources', 'Endpoints', and 'Business Information'. The main area is titled 'Resources' and shows a configuration for the resource '/books2016'. It includes fields for 'POST' and 'GET' methods, 'Consumes', 'Produces', 'Description', 'Nickname', and 'Return Type'. Below these is a 'Parameters' table with columns for Name, Type, Description, Required, and Data Type. The 'Authentication Type' dropdown is set to 'apiKey', and a dropdown menu is open showing 'apiKey', 'oauth2', 'basicAuth', and 'none'. A red arrow points to the 'none' option. Below the dropdown is an 'SLA' section with a table for SLA Plan Name and Rate Limit. The 'Enable Caching' checkbox is also visible.

Name	Type	Description	Required	Data Type
bookid	path		true	string

SLA Plan Name	Rate Limit
TRY OUT	10 Requests / MINUTE
UNLIMITED	1000 Requests / MINUTE

Create API – Select Post

The screenshot displays the API Manager Portal interface. On the left is a navigation sidebar with options: Back to Main, Basic Settings, Security (Authentication, Configuration, CORS), SLA Plans, Caching, Resources (highlighted), Endpoints, and Business Information. The main area is titled 'Resources' and shows a configuration for a resource at the path `/books2016`. A red arrow points to the `POST` method selected for the resource `/books2016/bookAdd`. The configuration includes:

- Consumes:** An empty text box.
- Produces:** `application/json`
- Description:** `empty`
- Nickname:** `getBookDetailsJSON`
- Return Type:** `Object`

Below the configuration is a 'Parameters' table:

Name	Type	Description	Required	Data Type
<code>bookid</code>	<code>path</code>		<code>true</code>	<code>string</code>

Other settings include:

- Authentication Type:** `none`
- SLA:** SLA Plan Name: `UNLIMITED`, Rate Limit: `1000 Requests / MINUTE`
- Enable Caching:**

At the bottom, a list of other resources is visible:

- `GET` `/books2016/{bookid}`
- `DELETE` `/books2016/{bookid}`

Create API – Confirm Authentication

The screenshot shows the API Manager Portal interface. The left sidebar contains navigation options: Back to Main, Basic Settings, Security (with sub-items Authentication and Configuration), CORS, SLA Plans, Caching, Resources, Endpoints, and Business Information. The main area is titled 'Resources' and shows a configuration for the resource '/books2016'. The selected operation is 'POST /books2016/bookAdd'. The configuration includes fields for Consumes, Produces, Description (set to 'empty'), Nickname (set to 'addBook'), and Return Type (set to 'Object'). A table of Parameters is shown below, with one parameter named 'NA' of type 'body', required, and of data type 'struct'. The 'Authentication Type' is set to 'apiKey', which is highlighted with a red box. At the bottom, the SLA section shows a Rate Limit of '10 Requests / MINUTE'.

Resources

Cancel Save Publish

/books2016 Add Operation Manage Models Mapped Name: /books2016

POST /books2016/bookAdd

Consumes Produces Description

Nickname Return Type

addBook Object

Parameters

Name	Type	Description	Required	Data Type
NA	body		true	struct

Authentication Type apiKey

SLA

SLA Plan Name Rate Limit

10 Requests / MINUTE

Create API – Save Changes

The screenshot shows the API Manager Portal interface. On the left is a navigation menu with options: Back to Main, Basic Settings, Security (Authentication, Configuration, CORS), SLA Plans, Caching, Resources (highlighted), Endpoints, and Business Information. The main area is titled 'Resources' and shows a configuration for a resource named '/books2016'. At the top right of this section are buttons for 'Cancel', 'Save' (with a red arrow pointing to it), and 'Publish'. Below this, there are buttons for 'Add Operation', 'Manage Models', and 'Mapped Name: /books2016'. The selected operation is a POST method for '/books2016/bookAdd'. The configuration includes:

- Consumes:** application/json
- Produces:** (empty)
- Description:** empty
- Nickname:** addBook
- Return Type:** Object

A 'Parameters' table is shown below:

Name	Type	Description	Required	Data Type
NA	body		true	struct

The 'Authentication Type' is set to 'apiKey'. An 'SLA' section shows two plans:

SLA Plan Name	Rate Limit
TRY OUT	10 Requests / MINUTE
UNLIMITED	1000 Requests / MINUTE

At the bottom, there is a checkbox for 'Enable Caching' and a preview of another resource configuration for a GET method on '/books2016/[bookId]'.

Test API - Get

The screenshot displays the API Manager Portal interface. The top navigation bar includes the 'API Manager Portal' logo and the user 'admin'. A left sidebar contains navigation options: Back to Main, API Details, Security (with sub-items Authentication, Configuration, and CORS), SLA Plans, Caching, Resources, Test this API, Endpoints, and Business Information. The main content area is titled 'Rest2016 > Try Out' and features an 'Edit' button. It lists three API resource groups: /books2016, /authors2016A, and /authors2016. Each group contains a list of endpoints with their respective HTTP methods. A red arrow points to the GET endpoint for /books2016/{bookid}.

Resource	Method	Endpoint
/books2016	POST	/books2016/bookAdd
	GET	/books2016/{bookid}
	GET	/books2016/{bookid}
	DELETE	/books2016/{bookid}
	PUT	/books2016/bookUpdate
	GET	/books2016/bookList
/authors2016A	GET	/authors2016A/{authorid}
	DELETE	/authors2016A/{authorid}
	PUT	/authors2016A/authorUpdate
	POST	/authors2016A/authorAdd
	GET	/authors2016A/authorsList
/authors2016	GET	/authors2016/{authorid}
	DELETE	/authors2016/{authorid}
	PUT	/authors2016/authorUpdate
	POST	/authors2016/authorAdd

Test API - Get

The screenshot displays the API Manager Portal interface. On the left is a navigation sidebar with options: Back to Main, API Details, Security (Authentication, Configuration, CORS), SLA Plans, Caching, Resources, Test this API (highlighted), Endpoints, and Business Information. The main content area shows the 'Rest2016 > Try Out' section with an 'Edit' button. The selected endpoint is `/books2016`. The method `GET` is selected for the path `/books2016/{bookid}`. A table lists parameters:

Name	Type	Description	Is required	Data Type
bookid	path		true	string

The `bookid` parameter is set to `6`. The 'Produces' dropdown is set to `APPLICATION/JSON`. Below the dropdown are expandable sections for Request Details, Response Details, and Response Body. At the bottom, other endpoints are listed: `GET /books2016/{bookid}`, `DELETE /books2016/{bookid}`, `PUT /books2016/bookUpdate`, and `GET /books2016/bookList`.

Test API - Get

API Manager Portal admin

Rest2016 > Try Out Edit

/books2016

POST ▶ /books2016/bookAdd

GET ▼ /books2016/[bookid]

Parameters

Name	Type	Description	Is required	Data Type
bookid	path		true	string

bookid *

Produces

APPLICATION/JSON

Run API Call ←

+ Request Details

+ Response Details

+ Response Body

GET ▶ /books2016/[bookid]

DELETE ▶ /books2016/[bookid]

PUT ▶ /books2016/bookUpdate

GET ▶ /books2016/bookList

Test API - Get

API Manager Portal admin ?

Caching
Resources
Test this API
Endpoints
Business Information

bookid	path	true	string
--------	------	------	--------

bookid*
6

Produces
APPLICATION/JSON

Run API Call

Request Details

URL : http://localhost:9100/rest2016/v1.0/books2016/6 ←
Method : GET
Accept : application/json
Cache-Control : no-cache
If-Modified-Since : 0
Pragma : no-cache
Params :

Response Details

Response Body

```
{
  "COLUMNS": [
    "BOOKID",
    "FIRSTNAME",
    "LASTNAME",
    "AUTHORID",
    "TITLE",
    "BOOKIMAGE",
    "BOOKDESCRIPTION",
    "ISSPOTLIGHT",
    "GENRE"
  ],
  "DATA": [
    [
      6,
      "Emily",
      "Kim",
      6,
      "The Look of Innovation",
      "book05.jpg",
      "How one woman struggles to make artistic designs that effectively sell new products. Is it fiction, or is it fa",
      "Fiction"
    ]
  ]
}
```

Compare Code - Get

```
<cffunction name="getBookDetails" access="remote" httpmethod="GET" returntype="any"
  restpath="{bookid}" produces="application/json, application/xml">
  <cfargument name="bookid" required="true" restargsource="Path" type="string">

  <cfinclude template="inc_tracking.cfm">

  <cfquery name="qBookDetails" datasource="#request.readDSN#">
    select b.bookid, a.firstname, a.lastname, a.authorid, b.title, b.bookimage, b.bookdescription,
      b.isspotlight, b.genre
    from app.books b, app.authors a
    where a.authorid = b.authorid
    and b.bookid = #arguments.bookid#
  </cfquery>

  <cfset temp = restThrow(throw="false", errorCode="200", message="OK", detail="Action completed
  successfully.", data="#serializeJSON(insertResults)#", sourcename="#getfunctioncalledname()#")>

  <cfreturn qBookDetails>
</cffunction>
```

Compare Code – Call Get

```
<cfhttp url="http://localhost:9100/rest2016/v1.0/books2016/24" result="restResult" method="get">
```

```
<cfhttpparam type="header" name="Accept" value="application/json" />
```

```
</cfhttp>
```

```
<cfdump var="#restResult#">
```

```
<cfhttp url="http://localhost:8500/rest/rest2016/books2016/24/.xml" result="restResult2" method="get">
```

```
</cfhttp>
```

```
<cfdump var="#restResult2#">
```

Test API - Post

API Manager Portal admin ?

Rest2016 > Try Out Edit

/books2016

POST **/books2016/bookAdd**

Parameters

Name	Type	Description	Is required	Data Type
structData	body		true	struct

Body *

```
[{"firstname":"Keen", "lastname":"Haynes", "bio":"Keen was raised in WV and served 24 years in the United States Marine Corps", "isspotlight": "Y"}]
```

Consumes

Nothing selected

APPLICATION/JSON

+ Request Details

+ Response Details

+ Response Body

GET **/books2016/[bookid]**

GET **/books2016/[bookid]**

DELETE **/books2016/[bookid]**

Test API - Post

The screenshot displays the API Manager Portal interface. On the left, a navigation menu includes options like Authentication, Configuration, CORS, SLA Plans, Caching, Resources, Test this API, Endpoints, and Business Information. The main area shows the configuration for a POST request to the endpoint `/books2016/bookAdd`. A table lists the parameters, with `structData` being a required body parameter of type `struct`. The request body is a JSON object with fields for authorID, title, bookdescription, isspotlight, genre, bookimage, and thumbnailimage. Below the configuration, the 'Run API Call' section shows the request details, including the URL, method, headers, and parameters. Two red arrows point to the URL and the `api_key` parameter. The response details section is collapsed, and the response body shows a JSON structure with a generated key and a list of values.

API Manager Portal admin ?

/BOOKS2016

POST `/books2016/bookAdd`

Parameters

Name	Type	Description	Is required	Data Type
structData	body		true	struct

Body *

```
{ "authorID": "1", "title": "Test Add", "bookdescription": "still testing add", "isspotlight": "Y", "genre": "testing", "bookimage": "test.jpg", "thumbnailimage": "testimagesm.gif" }
```

Consumes

APPLICATION/JSON

Run API Call

Request Details

URL : `http://localhost:9100/rest2016/v1.0/books2016/bookAdd`

Method : POST

Cache-Control : no-cache

Content-Type : application/json

If-Modified-Since : 0

Pragma : no-cache

Params :

`api_key : 5a860cb7cd17437385857a5e65c87a30`

Data : `[{"authorID": "1", "title": "Test Add", "bookdescription": "still testing add", "isspotlight": "Y", "genre": "testing", "bookimage": "test.jpg", "thumbnailimage": "testimagesm.gif"}]`

Response Details

Response Body

```
<STRUCT ID="1"><ENTRY NAME="1" TYPE="STRING">33</ENTRY><ENTRY NAME="GENERATEDKEY" TYPE="STRING">33</ENTRY><ENTRY NAME="values (1, 'Test Add', 'test.jpg', 'testimagesm.gif', 'still testing add', 'Y', 'testin
```

Compare Code

```
<cffunction name="addBook" access="remote" httpmethod="POST" returntype="any" restpath="bookAdd">
  <cfargument name="structData" type="struct" required="true" >

  <cfinclude template="inc_tracking.cfm">
  <cfinclude template="inc_auth.cfm">

  <cfquery name="qAddBook" datasource="#request.readDSN#" result="insertResults">
    insert into app.books (authorid, title, bookimage,thumbnailimage, bookdescription, isspotlight, genre)
    values (#structData.authorid#, '#trim(structData.title)#', '#trim(structData.bookimage)#',
    '#trim(structData.thumbnailimage)#', '#trim(structData.bookdescription)#', '#structData.isspotlight#',
    '#trim(structData.genre)#')
  </cfquery>

  <cfset temp = restThrow(throw="false", errorCode="200", message="OK", detail="Action completed
  successfully.", data="#serializeJSON(insertResults)#", sourcename="#getfunctioncalledname()#")>

  <cfreturn insertResults>
</cffunction>
```

Compare Code

```
<cfset variables.jsonStr = {authorID=1, title="Test Add", bookdescription="sstill testing add", isspotlight="Y",  
  genre="testing", bookimage = "test.jpg", thumbnailimage="testimagesm.gif"}>  
<cfset variables.dataString = SerializeJSON(variables.jsonStr)>  
  
<cfhttp url="http://localhost:9100/rest2016/v1.0/books2016/bookAdd" result="restResult" method="post">  
  <cfhttpparam TYPE="Header"  
    name="content-type"  
    value="application/JSON">  
  <cfhttpparam TYPE="header"  
    name="api_key" value="5a860cb7cd17437385857a5a65c87e30">  
  <cfhttpparam TYPE="Body"  
    name="structData"  
    value="#variables.dataString#">  
</cfhttp>  
  
<cfdump var="#restResult#">
```

Compare Code

```
<cffunction name="addAuthor" access="remote" httpmethod="POST" returntype="any"
  restpath="authorAdd">
  <cfargument name="firstname" required="true" restargsource="form" type="string">
  <cfargument name="lastname" required="true" restargsource="form" type="string">
  <cfargument name="isspotlight" required="false" default="N" restargsource="form" type="string">
  <cfargument name="bio" required="false" default="" restargsource="form" type="string">

  <cfquery name="qAddAuthor" datasource="#request.readDSN#" result="insertResults">
    insert into app.authors (firstname, lastname,bio,isspotlight)
    values ('#trim(arguments.firstname)#', '#trim(arguments.lastname)#',
'#trim(arguments.bio)#', '#arguments.isspotlight#')
  </cfquery>

  <cfreturn insertResults>
</cffunction>
```


API Manager Portal admin ?

PUT ▶ /authors2016A/authorUpdate

POST ▼ /authors2016A/authorAdd

Parameters

Name	Type	Description	Is required	Data Type
firstname	form		true	string
lastname	form		true	string
isspotlight	form		false	string
bio	form		false	string

firstname * lastname * isspotlight

bio

Consumes

←

[Run API Call](#)

Request Details

URL : <http://localhost:9100/rest2016/v1.0/authors2016A/authorAdd>
Method : POST
Accept : application/json
Cache-Control : no-cache
Content-Type : application/x-www-form-urlencoded
If-Modified-Since : 0
Pragma : no-cache
Params :
 api_key : 5a860cb7cd17437385857a5a65c87e30
Data : firstname=Bob&lastname=Jones&isspotlight=N&bio=Writes%20mainly%20science%20fiction%20novels&

Response Details

Compare Code

```
<cfhttp url="http://localhost:9100/rest2016/v1.0/authors2016A/authorAdd" result="restResult"
  method="post">

  <cfhttpparam TYPE="Header" name="content-type" value="application/x-www-form-urlencoded">
  <cfhttpparam TYPE="header" name="api_key" value="5a860cb7cd17437385857a5a65c87e30">
  <cfhttpparam TYPE="FormField" name="firstname" value="Keen">
  <cfhttpparam TYPE="FormField" name="lastname" value="Haynes">
  <cfhttpparam TYPE="FormField" name="isspotlight" value="Y">
  <cfhttpparam TYPE="FormField" name="bio" value="A retired United States Marine, Keen grew up in
  West Virginia. When not coding he can be found riding his Harley">

</cfhttp>

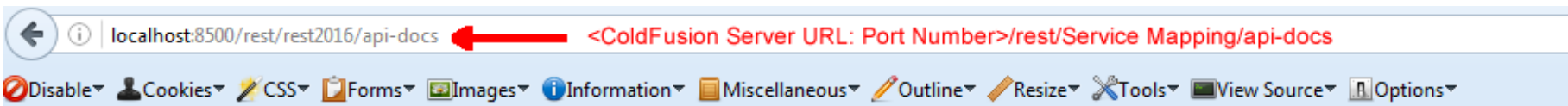
<cfdump var="#restResult#">
```

Generating Swagger Docs – Application.cfc

```
RDS Query Viewer  Application.cfc  authors2016.cfc  books2016.cfc  authors2016A.cfc
",, ', ab AB  try  [X] [$] /%# # <!-- //
1 <cfcomponent output="false">
2   <cfparam name="request.readsn" default="cfbookclub">
3   <cfset this.name = "bookclubDemo2016">
4   <cfset this.applicationTimeout = createTimeSpan(2,0,0,0)>
5   <cfset this.clientManagement = "true">
6   <cfset this.sessionManagement = "true">
7   <cfset this.loginStorage = "session">
8   <cfset this.sessionTimeout = createTimeSpan(0,0,20,0)>
9   <cfset this.setClientCookies = "true">
10  <cfset this.setDomainCookies = "true">
11  <cfset this.scriptProtect = "false">
12  <cfset this.restsettings.generateRestDoc="true">
13  <cfset this.restsettings.restDocInfo.title="CFSummit">
14  <cfset this.restsettings.restDocInfo.apiVersion="1.0">
15  <cfset this.restsettings.restDocInfo.description="These are the REST services used in CF Summit demo">
16  <!--- <cfset this.restsettings.restDocInfo.termOfServiceUrl="http://thermofisher.com">--->
17  <cfset this.restsettings.restDocInfo.contact="keen.haynes@thermofisher.com">
18  <!---<cfset this.restsettings.restDocInfo.license="adobe 1.0">--->
19  <!---<cfset this.restsettings.restDocInfo.licenseUrl="http://thermofisher.com">--->
20
21 <cffunction name="onApplicationStart" returnType="void" output="false">
```

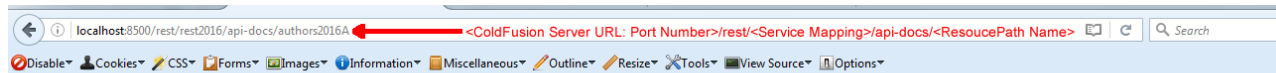
[API Manager Help / Generating Swagger documents](#)

Generating Swagger Docs – Resource Listing



```
{
  "apiVersion":"1.0",
  "swaggerVersion":"1.2",
  "apis":[
    {
      "path":"/authors2016"
    },
    {
      "path":"/authors2016A"
    },
    {
      "path":"/books2016"
    }
  ],
  "info":{
    "title":"CFSummit",
    "description":"These are the REST services used in CF Summit demo",
    "contact":"keen.haynes@thermofisher.com"
  }
}
```

Generating Swagger Docs – API Declaration



```
{
  "swaggerVersion": "1.2",
  "apiVersion": "1.0",
  "basePath": "http://localhost:8500/rest/rest2016",
  "resourcePath": "/authors2016A",
  "apis": [
    {
      "path": "/authors2016A/{authorid}",
      "description": "",
      "operations": [
        {
          "nickname": "getAuthorDetails",
          "method": "GET",
          "summary": "",
          "type": "Object",
          "produces": [
            "application/xml",
            "application/json"
          ],
          "parameters": [
            {
              "name": "authorid",
              "paramType": "path",
              "allowMultiple": false,
              "required": true,
              "type": "string"
            }
          ]
        }
      ],
      {
        "nickname": "deleteAuthor",
        "method": "DELETE",
        "summary": "",
        "type": "Object",
        "parameters": [
          {
            "name": "authorid",
            "paramType": "path",
            "allowMultiple": false,
            "required": true,
            "type": "string"
          }
        ]
      }
    ],
    {
      "path": "/authors2016A/authorUpdate",
      "description": "",
      "operations": [
        {
          "nickname": "updateAuthor",
          "method": "PUT",
          "summary": "",
          "type": "Object",
          "parameters": [
            {
              "name": "authorID",
              "paramType": "form",
              "allowMultiple": false,
              "required": true,
              "type": "string"
            },
            {
              "name": "firstname",
              "paramType": "form",
              "allowMultiple": false,
              "required": true,
            }
          ]
        }
      ]
    }
  ]
}
```

Swagger UI – Calling API Declaration

swagger

http://localhost:8500/rest/rest2016/api-docs/authors2016

Explore

authors2016

Show/Hide | List Operations | Expand Operations

DELETE /authors2016/{authorid}

GET /authors2016/{authorid}

Response Class (Status 200)

Object

Response Content Type: application/xml

Parameters

Parameter	Value	Description	Parameter Type	Data Type
authorid	(required)		path	string

Try it out!

PUT /authors2016/authorUpdate

POST /authors2016/authorAdd

GET /authors2016/authorsList

[BASE URL: /rest/rest2016]

Analytics Dashboard

The screenshot displays the API Manager Portal interface. The browser address bar shows the URL `localhost:9000/portal/#/browse`. The page title is "API Manager Portal" and the user is logged in as "admin".

The left sidebar contains the following navigation items:

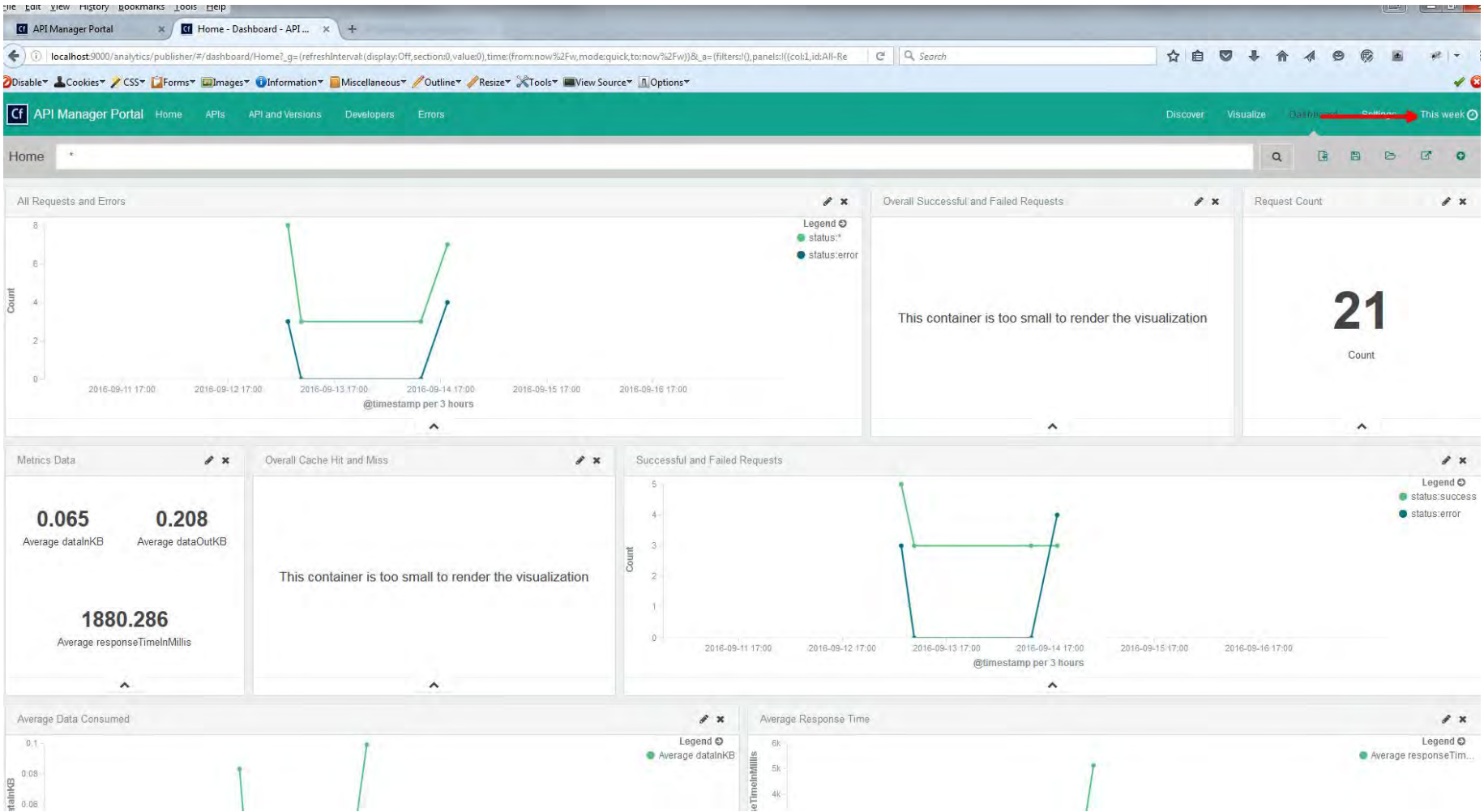
- My APIs
- Create API
- API Catalog
- Subscriptions
- Notifications
- Key Stores
- Analytics (highlighted with a red arrow)

The main content area shows a grid of API cards. At the top, there is a search bar labeled "Search APIs" and two dropdown menus: "All" and "Newest".

The API cards displayed are:

- rest2016** (DRAFT): VERSION : v1.1, VISIBILITY : public
- rest2016** (DRAFT): VERSION : v1.0, VISIBILITY : public
- mobileApp** (DRAFT): VERSION : v1.0, VISIBILITY : public
- summit** (DRAFT): VERSION : v1.0, VISIBILITY : public
- priceOrder** (DRAFT): priceOrder to OFM, VERSION : v1.0, VISIBILITY : public
- rfidCycleCount** (DRAFT): VERSION : v1.0, VISIBILITY : public
- MobileAuth**
- TestRest** (DRAFT): VERSION : v1.0, VISIBILITY : public
- TestMobile** (DEPRECATED): VERSION : v1.0, VISIBILITY : public

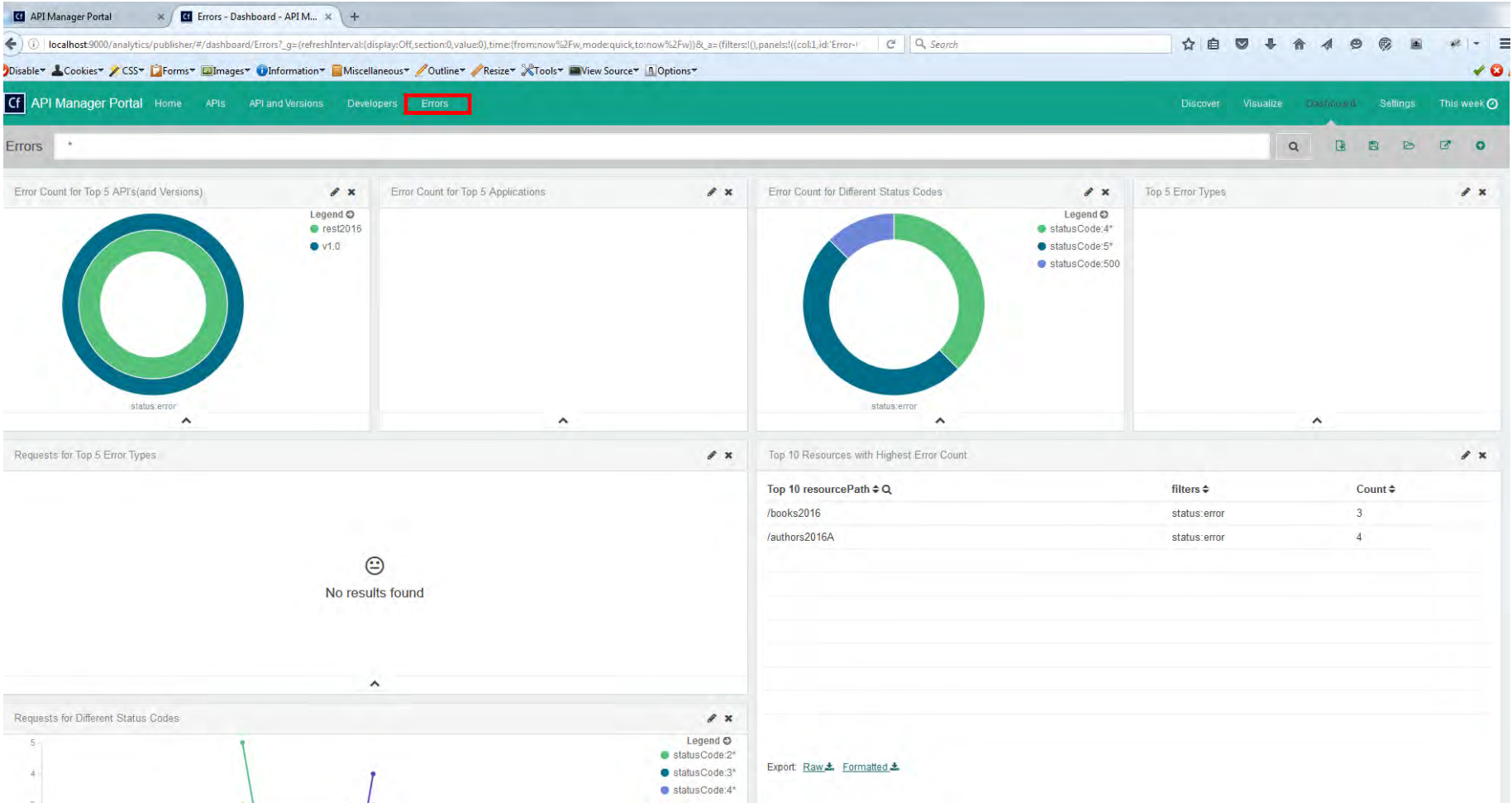
Analytics Dashboard - Home



Analytics Dashboard – Setting Time Frame

The screenshot shows the API Manager Portal Analytics Dashboard. The top navigation bar includes 'API Manager Portal', 'Home', 'APIs', 'API and Versions', 'Developers', and 'Errors' (highlighted with a red arrow). The dashboard features a 'Time Filter' section with 'Quick', 'Relative', and 'Absolute' categories. A grid of time filter options is displayed, including 'Today', 'This week', 'This month', 'This year', 'The day so far', 'Week to date', 'Month to date', 'Year to date', 'Yesterday', 'Day before yesterday', 'This day last week', 'Previous week', 'Previous month', 'Previous year', 'Last 15 minutes', 'Last 30 minutes', 'Last 1 hour', 'Last 4 hours', 'Last 12 hours', 'Last 24 hours', 'Last 7 days', 'Last 30 days', 'Last 60 days', 'Last 90 days', 'Last 6 months', 'Last 1 year', 'Last 2 years', and 'Last 5 years'. The main dashboard area contains several widgets: 'All Requests and Errors' (line chart showing status counts over time), 'Overall Successful and Failed Requests' (message: 'This container is too small to render the visualization'), 'Request Count' (card showing a count of 21), 'Metrics Data' (cards for Average dataInKB: 0.065 and Average dataOutKB: 0.208), 'Overall Cache Hit and Miss' (message: 'This container is too small to render the visualization'), and 'Successful and Failed Requests' (line chart showing success and error counts over time).

Analytics Dashboard - Errors



Helpful Links

REST RESOURCES

[Getting started with RESTful web services in ColdFusion](#)

[REST Web Services In ColdFusion](#)

[Generating Swagger Docs](#)

[Swagger Docs CFML](#)

[Enable CORS Support](#)

[Getting CORS To Work With Apache](#)

SOFTWARE

[Adobe ColdFusion](#)

[SoapUI](#)

[Postman](#)

[Boomerang](#)

[Swagger UI](#)

API MANAGER RESOURCES

[ColdFusion API Manager Overview](#)

[API Manager Features Summary](#)

[Adobe ColdFusion API Manager Administrator](#)

[Introduction to Adobe ColdFusion API Manager](#)

[API Administrator Portal](#)

[API Manager-Metrics and Logging](#)

[API Manager Analytics Dashboard](#)

[API Manager Authentication Types](#)

APPLICATION DEMO

QUESTIONS?