cornerstone
realize your potential

Reporting API

Starter Guide and Best Practices

Version 2.1

February 2020

1601 Cloverfield Blvd, Ste., 600S | Santa Monica, CA 90404 | 310.752.0200 | fax 310.752.0199

Table of Contents

Document History

Author	Version	Sections	Date	Changes Made
Product Manager	1.0	All	September 2017	Initial Document
Product Manager	1.1	5.7	October 2017	Modified section 5.7 on Unique IDs
Product Manager	1.2	5.1.1, 5.1.2, 5.4, 5.8, 7.1	June 2018	Following changes were made in this version: • Updated sections 5.1.1 and 5.1.2 to add more clarity to the Session Token and Service Call steps. • Updated section 5.4 to provide additional details on CSOD's implementation of the OData paging feature, streaming mechanism, record count feature and query performance considerations. • Added section 5.8 describing the various errors that can manifest in the Reporting API and the recommended steps to fix the errors. • Updated section 7.1 – Updated FAQs
Product Manager	1.3	5.4.4	July 2018	Modified JSON example in section 5.4.4 - Paging and Ordering: Impact on Performance.
Product Manager	1.4	2, 3.1, 7.1	November 2018	Following changes were made in this version: • Updated CSC community name for RTDW documentation • Add clarification regarding session token validity period in FAQ section
Product Manager	2.0	3.2, 4, 5.1, 6.1	July 2019	• Removed references to trials and STS authentication. • Added steps for installing Reporting API in Pilot/Stage in section 4. • Added information about OAuth 2.0 in section 3.2 and section 5.1. • Added link to Edge Answers in section 6.1.
Product Manager	2.1	3.2, 5.1	February 2020	• Added guidance around OAuth 2.0 scopes

TECHNICAL SPECIFICATION DOCUMENT

Abstract:

This document provides provide guidance regarding the setup and best practices for consuming CSOD's Reporting APIs

Created By:

Narayan Aier

1. Introduction

The Reporting API is a public facing web service that allows clients programmatic read-only access to their Cornerstone data via the Real-time Data Warehouse (RTDW). The API is RESTful, adheres to the OData protocol (http://www.odata.org/) and dynamically adjusts to reflect any client's schema. Currently, it allows access to all the reporting views in the report schema (report.vw_rpt_*), which are the data source for custom reporting in the Cornerstone application.

1.1. Document Summary

The purpose of this document is to provide guidance regarding the setup and best practices for consuming Cornerstone's Reporting APIs. Note that this document is a supplemental aid provided by Cornerstone. It is meant to augment the Reporting API documentation and the data warehouse documentation available in the API Explorer.

If you believe any of the information below is inaccurate or contains ambiguous information, please contact your Integration Consultant or log a case with Global Product Support (GPS).

1.1.1. Intended Audience

This document is intended for the client's technical team that will build the client's side of the integration with the Reporting APIs. It is expected that this technical team has prior experience with REST web services and the OData protocol. Business stakeholders or technical team members who may not have this knowledge may still refer to this document to get additional insights into the Reporting API, however, please direct any questions specific to the REST or OData protocols to your team members with this knowledge.

2. Product Documentation

While this document provides a good starting point to understand the Reporting API, you will need to reference two additional sets of documents while developing your code.

2.1. Data Warehouse

- Documentation describing the schema for the various views in the RTDW is available in the 'RTDW Documentation for Reporting API, RDW, and Data Exporter' community within the Cornerstone

Success Center (CSC). As you develop code and/or ETL to consume CSOD's Reporting APIs, you will need to refer to this documentation for schema details and data dictionary.

- Please note that when you review this documentation, only the views prefixed with 'vw_rpt' are available via the Reporting API.
- You can also post questions to the group in the CSC community and receive updates on patches and releases.

2.2. API Explorer

- Current documentation on the Reporting API endpoints, authentication mechanism, sample code, and supported OData queries is available via the portal in the API Explorer: *Admin* > *Tools* > *EDGE* > *API Explorer* > *Reporting API.*
- As a system administrator, if you cannot navigate to the API Explorer, please advise your Integration Consultant or submit a case to Global Product Support (GPS).
- If you wish to provide access to the API Explorer to your team members who do not have access to your CSOD portal, you can refer them to the publicly available link for the API Explorer. No permissions or backend settings are needed to access this link.

3. Considerations

3.1. Ongoing Maintenance and Testing

For information related to patches and releases, please check the Client Success Center to retrieve in-depth information on patch/release schedules as well as documentation outlining the upcoming changes. Specifically for the Reporting API, please monitor any changes regarding data warehouse as they are published in the 'RTDW Documentation for Reporting API, RDW, and Data Exporter' community. The release calendar available in the Client Success Center outlines the exact dates for each release and patch.

- If you cannot access the Client Success Center, please advise your Integration Consultant or Client Success Manager.

Best Practices

- Please note that during a release, changes are pushed to the staging environment (typically 3 weeks) prior to the push to production/pilot. It is highly recommended that you maintain a test environment before and after implementation to monitor for any breaks to existing API integrations.

3.2. Security

There are limited data access controls with the Reporting API. You can control the views your API keys have access to by adjusting the *scope* of the application. However, **there are no available constraints to grant/deny access at a field or data level**. For example, using OAuth 2.0 scopes, you can restrict access to your application to just the user and transcript views. However, there is no facility to further restrict access to data for users from specific division or region. This type of security must be managed by the client. As a result, it is very important to not share any API keys or codes to any individuals who should not have access to your data within CSOD.

Best Practices

- OAuth 2.0 Applications – Cornerstone's APIs use the OAuth 2.0 authorization framework. Currently, Cornerstone only supports OAuth 2.0's Client Credential grant type. You can adjust the *scope* of the application and the tokens. To get a client ID/secret, you must register your application by navigating to Admin > Tools > Edge > API Management > Register OAuth 2.0 Application > Register New Application. On this page, the user that you associate with the OAuth 2.0 application must have the 'Reporting API – Read Only' permission.
    - You can control access to the Reporting API by adding/removing the 'Reporting API – Read Only' permission to the user associated with the OAuth 2.0 application.
    - You can control access to specific endpoints in the Reporting API by adjusting the *scopes* for your application.
- You can also temporarily revoke access by disabling the OAuth 2.0 application in the API Management page. You can permanently revoke access by uninstalling the OAuth 2.0 application in the API Management page. Please note that this will immediately revoke any access tokens issued by Cornerstone for that application.

3.3. Reporting Environments

Every CSOD client typically has three environments – Pilot, Stage and Production. The Reporting API can be enabled in all three environments. Clients should ideally maintain a stage or test environment on their end to consume and test the Reporting APIs.

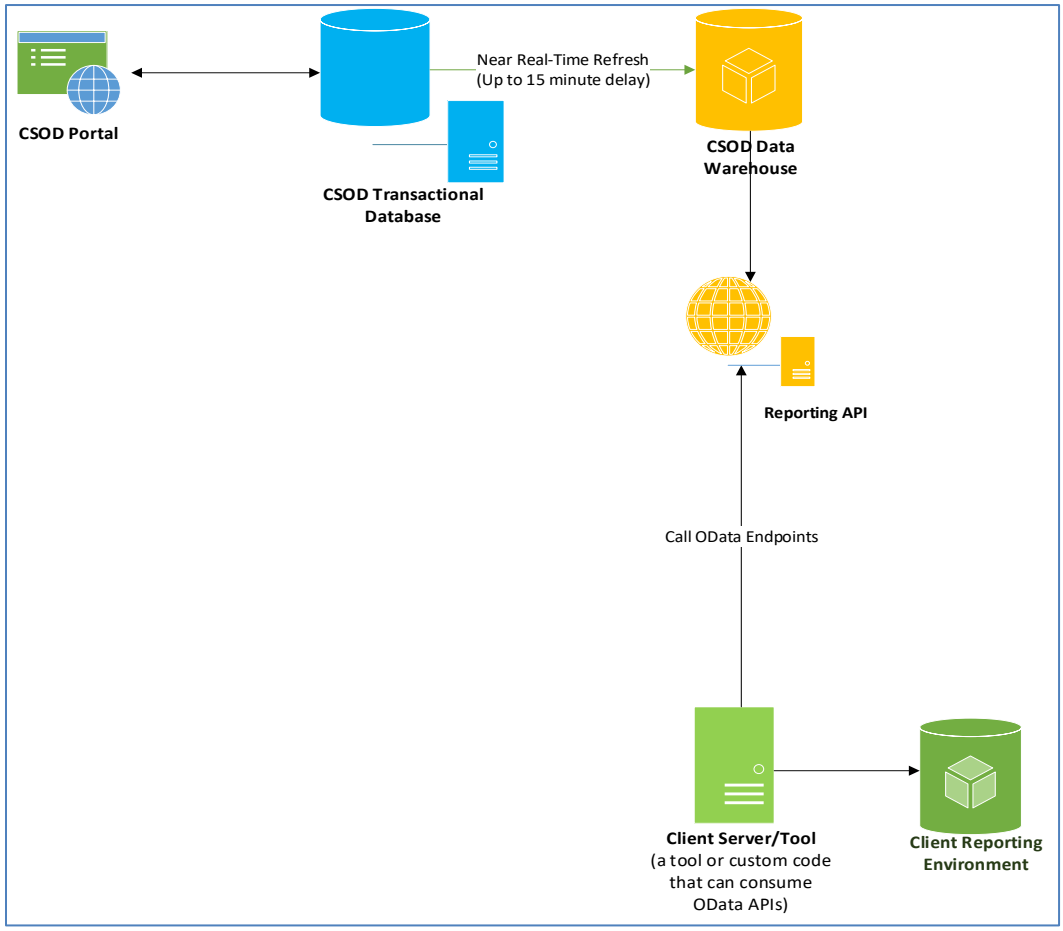
Best Practices

- Please check with your implementation team or system administrators to confirm which environment has the most relevant data for testing.

- It is recommended to use the CSOD pilot or staging environment while testing. This will provide the team the ability to stage data for validation purposes. **It is highly recommended not to stage any data in the production environment. This information CANNOT be deleted.**
- As previously indicated, it is also important for clients to create and maintain a test environment for release planning.
- It is important to note that the data warehouse may have certain database ID's (such as lookup IDs and form question IDs) that can be out of sync between environments. If possible, it is recommended to schedule a copy-down from production to pilot/stage for consistency with database ID fields. To schedule a copy down, please contact your Integration Consultant or log a case with Global Product Support (GPS). **Please inform all implementation teams or your Client Success Manager prior to scheduling a copy down as this could affect other work being done across environments.**

3.4. CSOD Data Warehouse

Every portal and environment has two sets of databases – a transactional database and a data warehouse database (officially referred as the Real-time Data Warehouse (RTDW)). The RTDW is refreshed near real-time from the transactional database. The diagram below illustrates this architecture.



Best Practice

- It is important to note that there could be up to a 15-minute delay between the syncing of the transactional database and the data warehouse. This should be noted as you build out any delta processes. For example, if you are using a date in a $filter condition, you may want to allow for a 15-minute window.

Sample Scheduling

Condition	Schedule
Where modified date >= 1:00 AM and < 2:00 AM	2:15 AM
Where modified date >= 2:00 AM and < 3:00 AM	3:15 AM
Where modified date >= 3:00 AM and < 4:00 AM	4:15 AM

4. Enabling Reporting API

There is no charge to access the Reporting API in the Pilot and Stage environments. Navigate to Admin > Tools > Edge > Marketplace. Search for and click on the **Reporting API**. Click the **Install** button to get started. Accept the terms and conditions and click on **Configure Now**. On the API Management page, ensure the toggle for Reporting API is turned on. Then, follow the steps below to get your API credentials.

For Production access, submit a purchase inquiry through the Marketplace. Navigate to Admin > Tools > Edge > Marketplace. Search for and click on the **Reporting API**. Click the **Purchase Inquiry** button to get started. Once you submit the inquiry, your Cornerstone Client Executive will reach out to you with next steps for completing the purchase and enablement process. Once that is completed, follow these steps to get your API credentials.

Registering your OAuth 2.0 Application

1. Log in to your portal. Ensure you have the 'Edge APIs - Manage' security permission.
2. Navigate to Admin > Tools > Edge > API Management > Manage OAuth 2.0 Applications.
3. Click on Register New Application. Review the Setup Instructions on that page to create your application. Note that the user associated with the OAuth 2.0 application must have the 'Reporting API - Read Only' security permission.
4. Copy the Client Secret and Client ID and use it in your external application that needs access to the Reporting API.

5. Consuming Reporting API

5.1. Authentication

Cornerstone's APIs use the OAuth 2.0 authorization framework. Currently, Cornerstone only supports OAuth 2.0's Client Credential grant type. You can adjust the *scope* of the token to the specific Reporting API endpoints you wish to access. More information about the OAuth 2.0 framework, the steps to register your application, get a client ID/secret, and make API calls is provided in the Getting Started page of the API Explorer.

Best Practices

- Please note that there is an expires_in field in the response of the OAuth 2.0 token endpoint. This field contains the validity period of the OAuth 2.0 access token issued by Cornerstone in seconds. It is not required to generate a new access token for every API call. You can reuse the same access token until the token remains active.
- You should limit the scope of your OAuth 2.0 application to just the endpoints you need access to. More information about OAuth 2.0 scopes can be found here: Access Control Using Scopes and Security Permissions.

5.2. OData Queries

CSOD's Reporting APIs use the OData protocol. For more information regarding the OData protocol, please check http://www.odata.org/. Cornerstone is currently using OData version 4.0. Please note that not every OData option is available. Please see the API Explorer for more details on the options you have with CSOD's implementation of OData.

Best Practices

- Reducing payload at source:
    - The Reporting API exposes views built by the CSOD reporting teams, which produce denormalized representation of the relational tables used by the CSOD application. As a result, the views can become very large vertically (number of rows), but more importantly they can be very wide horizontally by the number of properties or fields they cover. You can think of the views as an aggregation of not only the records in each table but also of the columns they hold.
    - It's important to review your use case and what you plan to do with the data, retrieve only what you need rather than getting everything.
    - It is much more efficient to reduce your payload at the source, doing so will yield significant performance gains in response and processing time from the service.
    - For this reason, it is always a best practice to utilize the **$select** option to eliminate any unnecessary fields.
- Most browsers and applications have a max URL of 2,083 characters. This is important to keep in mind as you build out your solution.

5.3. Joins

With the CSOD Reporting API, there is no option to perform any joins between views in a single service call. Because of this, there are two recommended solutions to bring in data from various views into a single report.

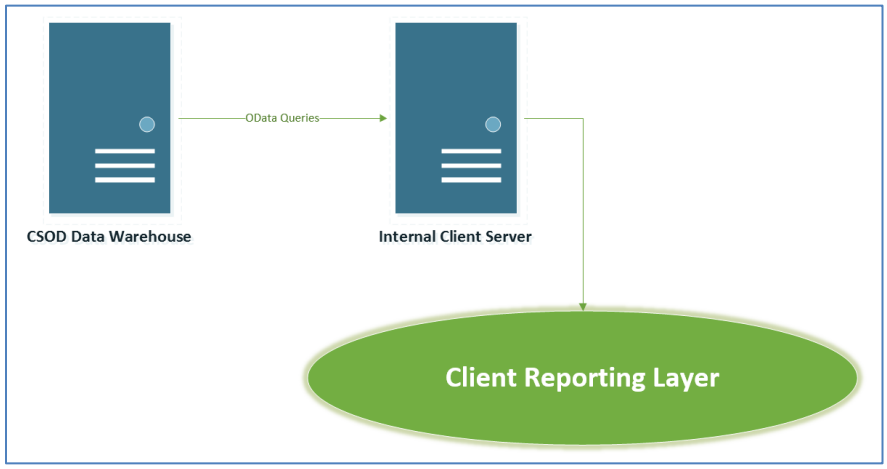
Option 1 – Iterative Loops

Within your coding environment, you can build out a looping mechanism to pull in information from multiple tables. In the below example, we want to pull applicant information as well as some data on the requisition they have applied to:

```
Endpoint = https://portal.csod.com/services/api/x/odata/api/views/vw_rpt_applicant
     For each job_applicant_id returned:
       Next Endpoint =
       https://portal.csod.com/services/api/x/odata/api/views/vw_rpt_recruiting?
                    $filter=ats_req_id eq { job_requisition_id from vw_rpt_applicant }
```

Option 2 – ETL to Internal Server

In this approach, you can pull various views into a local database in which you are able to create and manage your own queries, views, and security.



5.4. Paging

The Reporting API segments large results of data into pages. You can control the page size, i.e. the number of records per page, through the 'prefer' header in your request. For example,

```
prefer = odata.maxpagesize=2000
```

If the 'prefer' header is not included in your request, the Reporting API uses the **default page size of thousand records.**

5.4.1. Streaming

The Reporting API uses a streaming protocol, it sends data out to the client as soon as it becomes available rather than waiting until it receives all the records from the database before sending them out. This allows the API to provide access to very large data sets very efficiently, using a relatively small memory footprint on our servers, allowing us to serve many more simultaneous requests.

5.4.2. Record Duplication

To provide you the most up to date data, the Reporting API's data is synced from the transactional database on a frequency at most of fifteen minutes. This is important to consider because it means the data the Reporting API operates on can change at any time, while you are executing your queries.

As you retrieve records, page after page, newly synced data can cause the records positions to shift. A previously retrieved record can potentially re-appear in subsequent pages and potentially cause errors as you process it on your end.

It is highly recommended that you first stage the data you retrieve in a temporary storage, run any validation logic you deem necessary, including checking for duplicate records, before sending it to its final destination in your pipeline.

5.4.3. Record Count

Now that you are aware that the data may shift across pages while you are retrieving it, it is good practice to get a count of records at the beginning of your process to verify against at the end. The record count is a snapshot at the time of the request – it reflects the total possible records that meet the filtering criteria you specified. It is important to specify the same filtering criteria you will use when retrieving your records.

For example, using the following use case: Retrieve all completed transcript records in the last 24 hours.

1. Get the total record count.
   Request:

```
Get
https://{yourcorp}.csod.com/services/api/x/odata/api/views/vw_rpt_transcript?$filter=user
_lo_comp_dt ge cast('currentDate - 24 hours', Edm.DataTimeOffset)&$count=true&$top=0
```

It's important to specify $top=0 when retrieving the total count, this way the service only returns the count. Calculating the total count can be a very expensive operation on the system, especially when the view has very large record set.

2. Parse the response.

```
{
    "@odata.context":
    "https://.../services/api/x/odata/api/views/$metadata#vw_rpt_transcript",
    "@odata.count": 31379,
    "value": []
}
```

3. Proceed to retrieve the records
   Request:

```
Get
https://{yourcorp}.csod.com/services/api/x/odata/api/views/vw_rpt_transcript?$filter=user
_lo_comp_dt ge cast('currentDate - 24 hours', Edm.DataTimeOffset)&$select=transc_user_id,
tansc_object_id, reg_num, user_lo_score,user_lo_comp_dt
```

Response:

```
{
    "@odata.context":
    "https://.../services/api/x/odata/api/views/$metadata#vw_rpt_transcript",
    "value": [
    ],
    "@odata.nextlink":"https://{yourcorp}.csod.com/services/api/x/odata/api/views/vw_rpt_transcript?...&$skiptoken=..."
}
```

Repeat call @odata.nextlink for additional records.

4. Verify your accumulated record count to the initial count you retrieved
5. The record count can also be useful to determine what to set your page size to. If the total count is relatively low but higher than the default page size, you may elect to increase the page size by setting it higher to reduce the number of pages you go through. **You must use caution when increasing the page size, do not set it to an excessively large number as it could cause the request to timeout on the server.**

5.4.4. Paging and Ordering: Impact on Performance

The Reporting API implements the OData server driven paging protocol. When the following condition is met: *total count > page size*, the service will inject into the page response a link to retrieve the next page of results.

```
{
        "@odata.context":
        "https://{yourcorp}.csod.com/services/api/x/odata/api/views/$metadata#vw_rpt_transcript",
        "value": [ …],
        "@odata.nextLink":
        "https://{yourcorp}.csod.com/services/api/x/odata/api/views/vw_rpt_transcript?..."
}
```

"@odata.nextlink" should be treated as opaque, parsing it or manipulating could result in incorrect results and even errors.

Best Practices

The Cornerstone engineering team continuously works on tackling the performance challenges of the Reporting API and optimizations are frequently released which translate to better response times. To take advantage of these optimizations you should do the following:

- Adhere to the OData server driven paging protocol by using the "@odata.nextlink" to page through results
- **Refrain from using $orderby clause in your request.**

- To provide the best possible performance, the query engine leverages primary key fields and indexes of the underlying views. It does so by sorting (ordering) and filtering on those fields and therefore yielding improved results. Not all the columns in a view are indexable, doing so is resource prohibitive on the database server. When $orderby clause is specified it nullifies the optimizations in the query engine, yielding less performant results. On large datasets, this degradation in performance can be very significant.
- **If you do need to sort your results, we recommend you do so once you have the records on your end.** You would be operating on smaller sets of data, also as previously mentioned in this document, staging the data in temporary storage, before committing it to its final destination, gives you the opportunity to perform such operations

5.5. Custom Fields

5.5.1. Encoded Field ID

Whenever a custom field is created in the portal, it is assigned a database-generated encoded ID. **Please note this encoded ID may not be the same between environments.** The ID will never change within its respective environment except for copy downs.

The general rule for locating custom fields is by looking for '_cf' appended to a view name. **Please note that custom fields are categorized by type/entity and will not be found in a single view.** You can find the custom field type/entity in Custom Field Administration by navigating to: *Admin > Tools > Core Functions > Custom Field Administration.*

The custom field views include the encoded field ID and the value selected for that field. In order get the mapping between the encoded field ID and the name of the custom field as seen in your CSOD portal, you will need to log a case with Global Product Support (GPS). GPS can provide an extract that maps the ID to the field name.

5.5.2. Dropdowns, Multiple Checkboxes, and Branched Hierarchies

For dropdown, radio button, multiple checkbox, and branched hierarchy custom fields, the transactional view may provide you with an ID in your return. To map the ID with the description of the record, you must map the ID to one of the local views.

Example:

We have a 'Gender' user custom field labeled **user_custom_field_00001** that returns a value of 1 or 2. In the **vw_rpt_custom_field_value_local** view, $filter= cfvl_value_id eq 1 or $filter= cfvl_value_id eq 2 may return 'Male' or 'Female'.

Please note that these specific views are localized, meaning you may see multiple values for a given ID. This allows us to report on certain culture ID's or languages. If you are only looking for a single value, you will want to limit your query to a single culture ID (culture_id 1 is en-US).

Below is a listing of where to look up these values based on custom field type:

Custom Field View Name	Value Lookup View	Value ID
vw_rpt_application_cf	vw_rpt_custom_field_value_local	cfvl_value_id
vw_rpt_jp_incumbent_smp_cf	vw_rpt_custom_field_value_local	cfvl_value_id

Custom Field View Name	Value Lookup View	Value ID
vw_rpt_jp_ou_cf	vw_rpt_custom_field_value_local	cfvl_value_id
vw_rpt_jp_position_smp_cf	vw_rpt_custom_field_value_local	cfvl_value_id
vw_rpt_jp_successor_smp_cf	vw_rpt_custom_field_value_local	cfvl_value_id
vw_rpt_offer_letter_cf	vw_rpt_custom_field_value_local	cfvl_value_id
vw_rpt_ou_cf2	vw_rpt_custom_field_value_local	cfvl_value_id
vw_rpt_resume	vw_rpt_resume_section_attribute_value_local	rsal_attribute_id
vw_rpt_succession_incumbent_smp_cf	vw_rpt_custom_field_value_local	cfvl_value_id
vw_rpt_succession_successor_smp_cf	vw_rpt_custom_field_value_local	cfvl_value_id
vw_rpt_training_cf	vw_rpt_lo_form_cf_display_value_local	cfvl_field_id
vw_rpt_transaction_cf	vw_rpt_transaction_custom_field_option_local	tol_value_id
vw_rpt_transcript_cf	vw_rpt_lo_form_cf_display_value_local	cfvl_field_id
vw_rpt_user_cf	vw_rpt_custom_field_value_local	cfvl_value_id

5.6. Delta Conditions

For clients pulling information on a regular basis, it is recommended that you apply a **$filter** to your OData call. Depending on a given view, there may not be a last modified date for the information presented. Based on certain views, there may be other information or date filters that you can apply to limit the amount of information being returned. During the design phase of your reporting development, it is recommended to determine the best criteria that makes sense for your report specifications.

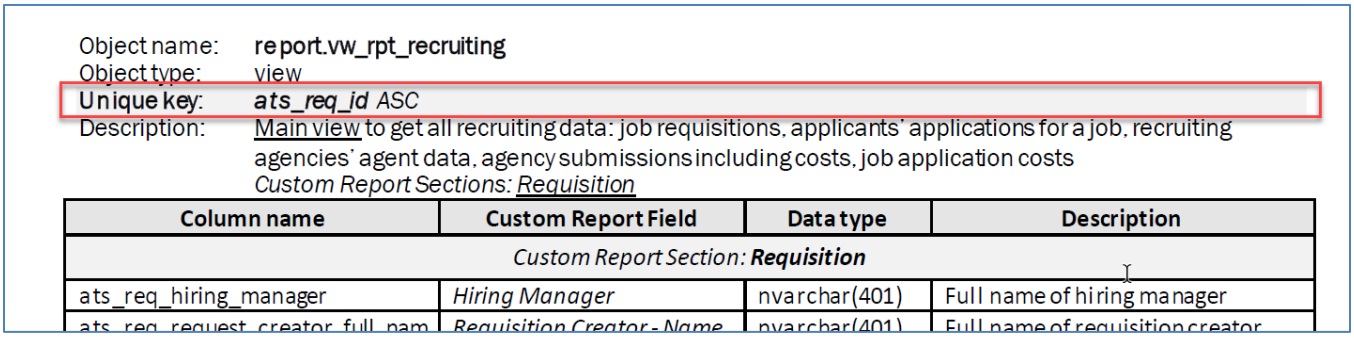
Example:

By applying the following conditions to the vw_rpt_transcript view, you should be able to get the most recent records since the last time you ran your query. While this condition will work for most scenarios, it is recommended that you evaluate your use cases and determine the most accurate delta conditions for your needs.

```
user_lo_create_dt > {date criteria} OR
user_lo_reg_dt > {date criteria} OR
user_lo_start_dt > {date criteria} OR
user_lo_comp_dt > {date criteria} OR
user_lo_last_access_dt > {date criteria} OR
user_lo_removed_dt > {date criteria} OR
user_lo_assigned_dt > {date criteria}
```

5.7. Unique IDs

The data returned via the Reporting API end points are from views in the CSOD data warehouse that are in turn created from CSOD's transactional database. Some views may not have a single unique ID for the entire dataset. The unique keys are identified in the RTDW documentation. As illustrated below, the unique keys are notated above each object.



Object name: **report.vw_rpt_recruiting**
Object type: view
Unique key: ***ats_req_id*** *ASC*
Description: Main view to get all recruiting data: job requisitions, applicants' applications for a job, recruiting agencies' agent data, agency submissions including costs, job application costs
Custom Report Sections: Requisition

Column name	Custom Report Field	Data type	Description
Custom Report Section: ***Requisition***			
ats_req_hiring_manager	*Hiring Manager*	nvarchar(401)	Full name of hiring manager
ats_req_request_creator_full_nam	*Requisition Creator - Name*	nvarchar(401)	Full name of requisition creator

5.8. Request Resiliency and Retry Guidance

The CSOD business objective for the Reporting API is to provide clients access to their growing reporting data programmatically. The goal for CSOD is to deliver a solution that not only meets the functional business needs of the clients, but one that is robust, secure and scalable. However, robust and scalable does not mean that errors will never occur. You should always code defensively, expect transient errors to occur and handle them gracefully.

In general responses with a status of 2xx are successful, however there are circumstances where this is not always the case. We will cover the most common types of errors and the special case when 200 is not actually successful.

5.8.1. Client Errors 3xx

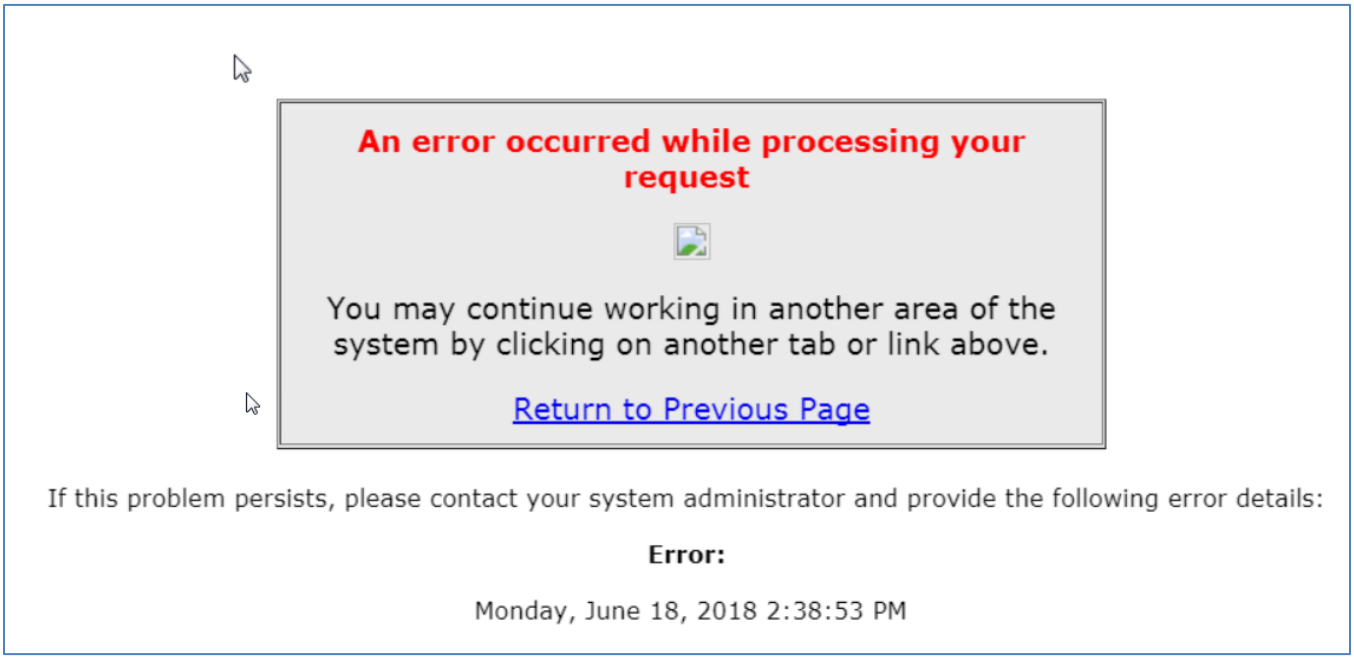
The most common case for this type of status is when a URL is incorrectly formatted and results in a redirect to the CSOD standard error page.

302 Redirect

The typical cause for a 302 error is appending query string parameters without specifying the delimiter that indicates the beginning of query string parameters. For example,

- Incorrect request, missing the delimiter: .../services/api/x/odata/api/views/vw_rpt_...**&**$filter=…
- Correct request, using the correct delimiter: ../services/api/x/odata/api/views/vw_rpt_...**?**$filter=...

This results in our servers issuing a 302 redirect to the error page. If your code is setup to follow redirects you will be redirected to the following error page.



5.8.2. Client Errors 4xx

The following errors require attention on your side, it usually involves fixing some aspects of your request before re-issuing it.

400 Bad Request / validation

Incorrect URL or parameters, such as the incorrect spelling of odata keywords $select, $filter, etc.

401 Unauthorized / Unauthenticated

Missing the right credentials or using an expired token in your request. Check the values you are sending and correct them accordingly.

403 Access Denied

The user account does not have the correct privileges to access the resource requested. If you encounter this error, you will need to login into your portal and assign the correct permissions to the user account being used to make the API calls.

404 Not Found

This error occurs when an incorrect resource is requested or the predicate for a resource does not match a record. For example,

- Incorrect resource: requesting vw_rpt_usr instead of vw_rpt_user
- Incorrect predicate: requesting vw_rpt_user/1234, when a user record with that id does not exist

429 Limit Exceeded

This is an indication that you have exceeded your throttling limit for the API.

Implement a back off logic accordingly to stagger your calls around the limits.

{

```
    "status": "429",
    "timeStamp":"2018-05-15T18:06:55+0000",
    "error": {
        "errorId": "77ed61e0-7052-4fad-b265-c52679ab2cac",
        "message": "CSOD Too many requests.",
        "code":"429",
        "description": null,
        "details": null
    }
}
```


5.8.3. Server Errors 5xx

Server errors are generally the result of a malfunction that occurred on our servers. In most cases they are transient errors and the recommendation is to retry the request until it succeeds. You should cap the number of retries after which you would error out and report the error to Global Product Support (GPS).

500 Server Error

An error occurred which may be transient. You should examine the details of the error :


```
{
    "status": "500",
    "timeStamp":"2018-05-15T18:06:55+0000",
    "error": {
        "errorId": "77ed61e0-7052-4fad-b265-c52679ab2cac",
        "message": "error message",
        "code":"xxx",
        "description": null,
        "details": null
    }
}
```


In some cases, the error could be due to a timeout on the server. This could be due to a couple of reasons:

- The load on the data source resulted in an excessive time before it responded to the request
- The page size requested was too large, resulting in a timeout between the data source and the service

In both cases retrying the request could succeed, however to increase your rate of success, it is recommended you implement a back-off logic when retrying.

You might want to progressively reduce your page size. Using the count feature could be helpful in making that determination. If the count returned indicates you are dealing with a very large number of records, for example, over a million records and the records are very wide, i.e. have a large number of columns, then you should reduce the page size and exercise the optimizations suggested earlier in this document.

503 Service Unavailable

This likely to occur when the system is down for maintenance. Cornerstone sends out notifications to our clients in advance in which maintenance windows are specified. The recommendation is to pause your process until after the maintenance window has elapsed. If you receive this error outside an announced maintenance window, you should notify Global Product Support (GPS) immediately so they can investigate.

5.8.4. Special Case: 200 not a Success

As stated in section 5.4.1 of this document, the Reporting API uses a streaming protocol to allow it to support large volumes of data. The results are sent back as they become available from the database. You can think of it as a fire hose – the data flows directly from the source, rather than being accumulated entirely into a container, then shipped as a whole. Errors are handled the same way as well.

This can sometimes cause the Reporting API to initially send back a 200 success response, however as you continue to page through additional records, it may timeout.

How would this error manifest itself?

Let's say you issue a request with a page size of 10,000 records. The server processes the query successfully and begins to send back results. Because a number of records are successfully sent, the response status is automatically set to 200 OK.

You continue to receive valid records, however at some point an internal server error occurs due to a resource exhaustion, this typically manifests itself as a timeout. The HTTP protocol does not allow changing the status of a response once it has been sent back, and therefore the client side is unaware that an error occurred.

On the receiving end, you would continue to process the records, unfortunately the data in the stream no longer represents a valid record. Your JSON parser errors out when it encounters data which is not consistent with a record. The response may look as follows:

```
{
              "@odata.context":
              "https://{yourcorp}.csod.com/services/api/x/odata/api/views/$metadata#vw_rpt_tr
              anscript",
              "value": [
              {valid records}
              error message, could be html
```

If you are collecting the entire JSON payload before you begin parsing it, then it is likely it is not valid JSON anymore. If you are parsing the data stream on the fly then you will run into the error in real-time.

In both scenarios, the error is most likely transient, and the recommendation is to follow the same approach as a 500 service error of retry with back-off logic.

6. Appendix

6.1. Frequently Asked Questions

Please search through our repository of FAQs in Edge Answers.

6.2. Acronyms and Abbreviations

API - Application Programming Interface - set of subroutine definitions, protocols, and tools for building application software.

CSC – Client Success Center

CSM – Client Success Manager

CSOD – Cornerstone OnDemand

ETL – Extract, Transform, Load – refers to a process in database usage and especially in data warehousing.

GPS – Global Product Support

REST – Representational State Transfer or RESTful web services allows systems to access and manipulate textual representations of web resources using a uniform and predefined set of stateless operations.

RDW – Replicated Data Warehouse

RTDW – Real-time Data Warehouse

_cf – When seen in a view name, this notation typically outlines custom fields within a specific entity

_local – When seen in a view name, this notation typically outlines a localized view providing language translations for a given field name (typically defined by culture_id)

_id – Commonly seen in the database views. This typically refers to a database ID.

_ref – Commonly seen in the database views. This typically refers to an ID located in the portal (Ex. User_ref = User ID)

6.3. Disclaimer

The information presented in this document is proprietary to Cornerstone OnDemand Inc. Please do not share this document with any other third parties without proper permission