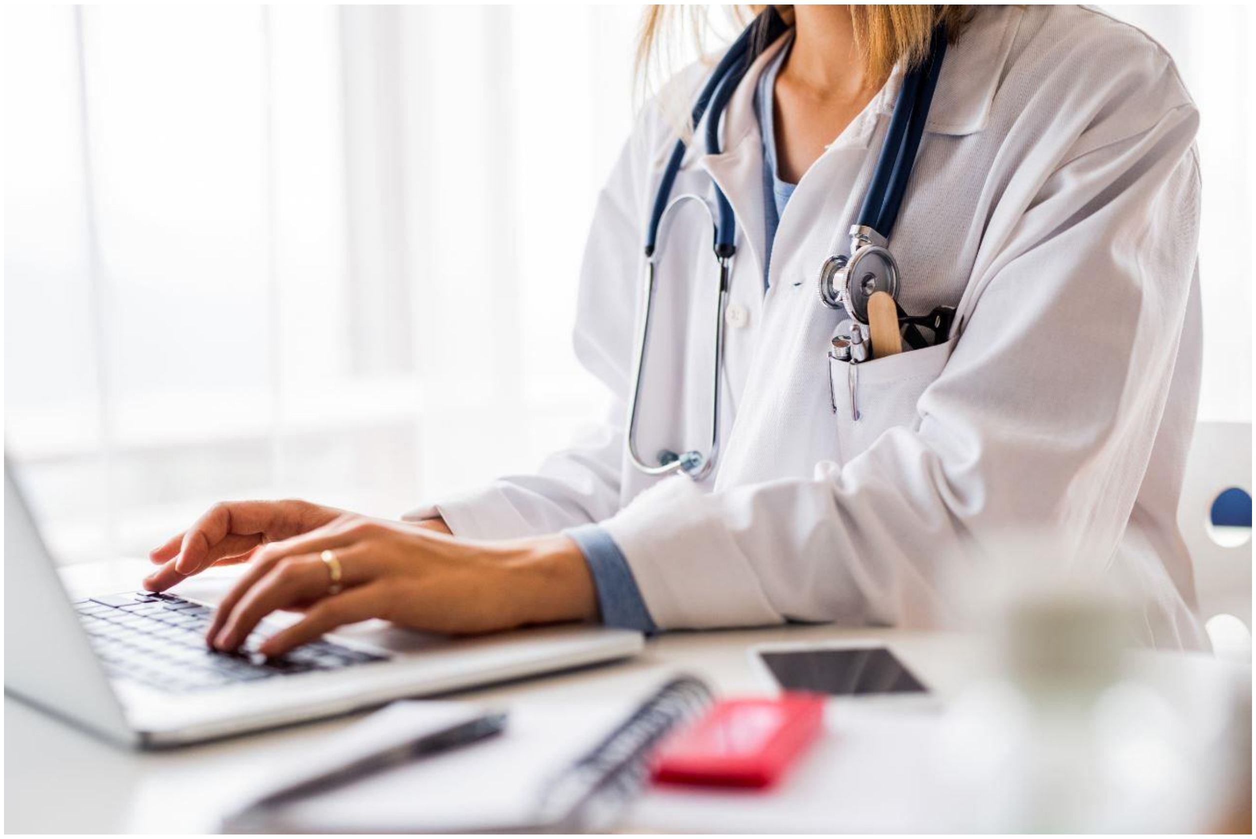
Ontario Health



ONE Access Viewlet Framework

Developer Guide

March 2021
Version 1.7

Document Control

The electronic version of this document is the only recognized version.

Approval History

Approver(s)	Title/Department	Approved Date

Revision History

Version No.	Date	Summary of Change	Changed By
0.4	2020-02-19	First draft of document provided to OntarioMD to support consent override for the EMR DHDR/DHIR integration project.	Ontario Health
1.2	2020-07-30	Initial trial-for-use document submitted for open review.	Ontario Health
1.3	2021-02-08	Draft publication for internal review, includes updates from open review comments.	Ontario Health
1.4	2021-02-10	Updates to section 10.1, Appendix B and C regarding UAO switching.	Ontario Health
1.5	2021-02-12	Updates to sections A.1 through A.4 and other revisions from Communications.	Ontario Health
1.6	2021-02-17	Version presented to Business Technical Committee.	Ontario Health
1.7	2021-03-15	Updates to section 7.0, 8.0 and Appendix C. Final version presented for Strategic Committee approval and publishing to corporate website.	Ontario Health

Document Sensitivity Level

Low

TABLE OF CONTENTS

1.0 Introduction

The Viewlet Framework is a common framework that defines the integration platform used to build reusable web applications, collectively referred to as **ONE Access Viewlets**. These Viewlets can be easily integrated with point of service (EMR, HIS's, etc.) and patient-portal solutions. ONE Access Viewlets address the need for small fit-for-purpose modern web applications that are both SPAs (single-page applications) and PWAs (progressive web applications). They can be used in a consistent modular fashion acting together to provide clinical value. As the number of reusable ONE Access Viewlets grows, they will be managed through a ONE Access Viewlet catalogue. This will allow clinical and patient applications to be developed with less effort, appropriate security, and a consistent user interface while following the same standards. Effectively, ONE Access Viewlets extend the capabilities of point of service (PoS) and patient-portal applications with functionality that would otherwise need to be developed and maintained, and conformance tested separately by each application.

This channel of access, consumption, and integration is enabled through the HL7® FHIR®, SMART on FHIR®, and FHIRcast specifications and includes a set of common UI elements, libraries, and standard services. As a result, it enables the ONE Access Viewlets to: function independently of each other or integrate with each other; always be up-to-date with the latest patches/fixes; be able to run on desktops and/or mobile devices; be integrated with any PoS or patient applications that adopt the Viewlet Framework published by Ontario Health.

2.0 Scope

The initial release of this guide aims to provide system developers of PoS applications with guidance on how to integrate with existing Ontario Health Viewlets, e.g. Provincial Consent Override Interface (PCOI) Viewlet.

The next release of this document will expand to include the development of new Viewlets.

3.0 Supporting Documentation

The following set of supporting documents are either referred to explicitly throughout this guide or can provide additional background information:

- Ontario Health's ONE® ID OpenID® Connect™ specification (abbreviated to **OIDC specification** from here on) for further details about ONE ID OAuth 2.0 tokens and their respective attributes mentioned in this guide.
- The Ontario Context Management System (CMS) - FHIR specification (abbreviated to **CMS iGuide** from here on) for a deep dive into the various CMS events, profiles and transactions mentioned here along with more sample JSON messages.
- The ONE Access Gateway Transport specification (abbreviated to **OAG specification** from here on) describes how client applications interact with Ontario Health's RESTful FHIR APIs including security with ONE ID OAuth 2.0 tokens and details on the structure of the HTTP headers discussed here.

- The ONE ID Provincial Identity Federation - SAML Interface specification (abbreviated to **SAML specification** from here on) as a resource to vendors of existing client applications wanting to integrate with the Viewlet Framework and currently use ONE ID SAML tokens.

4.0 Security

The Viewlet Framework falls within the domain of Ontario Health's EHR. Therefore, client applications integrating with the Viewlet Framework must adhere to the relevant guidelines and principles as outlined in the agency's EHR Security Toolkit.

5.0 Certified Web Browsers

Whenever possible, the latest versions of modern web browsers should be used to render Viewlets. If the latest versions cannot be used, Ontario Health recommends the following minimum versions based on testing:

- Google Chrome with minimum version 88.0.4324.96
- Mozilla Firefox with minimum version 84.0
- Apple Safari with minimum version 13.0.5
- Microsoft Edge with minimum version 87.0.664.60

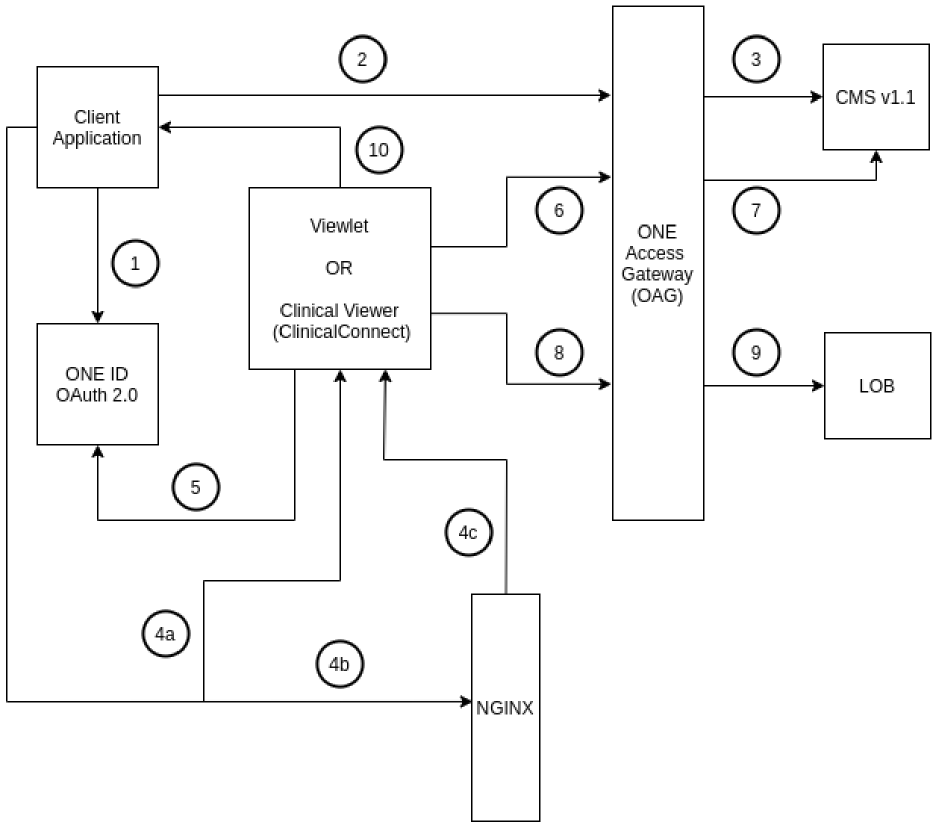
Rendering Viewlets in Chromium and Chromium-based browsers (e.g. JxBrowser) were not tested, but Ontario Health recommends using Chromium version 84 or higher.

6.0 Development Considerations and Limitations

Vendors of client applications should account for the following considerations and limitations for the first release of the Viewlet Framework:

- CMS v1.1 calls could be asynchronous, but the response must be tracked and error handling must occur appropriately and is the responsibility of the client application.
- Before invoking a Viewlet such as the PCOI Viewlet outlined in Appendix B, or launching any CMS-integrated clinical viewer such as ClinicalConnect outlined in Appendix C, all pre-requisite calls must first complete successfully.
- The current release of CMS does not implement event notifications as defined by FHIRcast's bi-directional publish/subscription (pub/sub) model which recommends the use of callback/postback URLs, webhooks and WebSockets. This is planned for the next release of CMS.
- The client application must be capable of launching a browser and performing all browser activities, including activities across tabs if required, in the same browser session.
- Viewlets can be opened in new windows or tabs, but some Viewlets may specifically call for the use of iFrames such as the PCOI Viewlet outlined in Appendix B.

7.0 Conceptual Flow Diagram



Steps:

1. The client application logs into ONE ID OAuth 2.0.
2. The client application builds the context object by writing to CMS, via the Provider ONE Access Gateway (OAG), over a series of events (login, patient selection, language change, etc.).
3. The Provider OAG validates the client application's ONE ID OAuth 2.0 token and passes the events to the CMS v1.1 server.
4.
    a. The client application launches ClinicalConnect™ or another clinical viewer by passing the following query parameters in the clinical viewer URL: *authzid* as *inheritanceID*, *hub.topic* as *launch*, and *FHIR_iss* as *iss*, and *ONE ID Identity Provider URL* as *SelectedIDP*; or

    b. The client application, via a call to an NGINX web server, launches the Viewlet in an iFrame by passing the following query parameters in the Viewlet URL: *authzid* as *inheritanceID*, *hub.topic* as *launch*, and *FHIR_iss* as *iss*; and then

    c. The NGINX server then returns the packaged source code to the browser to be run.
5. The Viewlet or clinical viewer logs into ONE ID OAuth 2.0.
6. The Viewlet or clinical viewer calls the CMS FHIR interface, via the Provider OAG, to get the context using the *contextGUID*.

7. The Provider OAG validates the Viewlet or clinical viewer's ONE ID OAuth 2.0 token and then passes the transaction to the CMS v1.1 server.
8. The Viewlet makes a call to a line of business (LOB) via a published API on the Provider OAG.
9. The Provider OAG validates the Viewlet's ONE ID OAuth 2.0 token and passes the call to the LOB.
10. The Viewlet lets the parent frame know it completed the transaction.

8.0 Use Cases Sequence Diagram

Each use case described in the following sections of this guide corresponds to an interaction operator in the UML sequence diagram with the same name. It is recommended to read the use cases and sequence diagram together. Due to the size of the diagram, it is posted online for readability and can be accessed via the link below.

Latest Online Sequence Diagram

Tip 1: For easier reading, in the top, right-hand corner, click on the *View menu* then select *Participants Overlay*.
Tip 2: To access the hyperlinks in the diagram, in the top right-hand corner, click on the *View* menu then deselect *Read Only Presentation Mode*.

9.0 Login Use Cases

The client application login flow outlined in the sequence diagram covers three separate use cases which are described here in full detail.

9.1 Authorization Use Case

1. The practitioner logs into the client application:

   Case 0: The client application integrates with ONE ID SAML (e.g., to access eConsult*)* and ONE ID OAuth 2.0 (e.g., to access DHDR and DHIR) with the ability to select one or more providers, also referred to as UAOs (Under Authority Of) using a UAO picker*.*

   Case 1: The client application integrates with ONE ID OAuth 2.0 with the ability to select one or more UAOs using a UAO picker.

   Case 2: The client application integrates with ONE ID OAuth 2.0 with a single UAO, either selected with a UAO picker or resolved ahead of time through its own means.

2. When making an initial authentication/authorization call to ONE ID, the client application requests an *id_token* and *access_token* to communicate with CMS v1.1.
   a. For authentication, i.e. *id_token*, request the following scope:
      - *openid*
   b. For authorization, i.e. *access_token*:
      i. To read and write to the CMS, request the following scopes:

        - *user/Context.read*
        - *user/Context.write*
    ii. To query line of business FHIR APIs (e.g. PCOI, DHDR, DHIR, etc.), request LOB scopes as needed.
    iii. To read URL endpoints for PCOI, CMS and Provider OAG, request the following scope:
        - *toolbar*
    iv. To share the client application's access privileges with other applications, request the access inheritance scope:
        - *azs*
    v. To select a specific UAO, pass a parameter called *uao* in the access token request with the value set to OID:UPI.

3. To process the ONE ID OAuth 2.0 token response, the client application will need to extract the following attributes from the token so they can be used in subsequent calls:
    a. *uao* - this can be retrieved by parsing through the *access_token* or *id_token* values in the token response.
    b. *authzid* - to be used for authorization *inheritanceID.*
    c. *toolbar* (Base64 encoded) parameters:
        i. *hub.url* - the URL for the CMS server,
        ii. *pcoi_url* - the URL for the PCOI Viewlet,
        iii. *FHIR_iss* - the URL for the FHIR issuer endpoint, which is the Provider ONE Access Gateway in this case.

    Parse out all key-value pairs in the JSON object of the *toolbar* attribute.

    For example:

```
{
  "hub.url":"https://cms.ontariohealth.on.ca/fhir4/cms",
  "pcoi_url":"https://pcoi.viewlet.ehealthontario.ca",
  "FHIR_iss":"https://providergateway.ehelthontario.ca"
}
```

9.2 createHubTopic Use Case

Support for different context sessions across multiple tabs will depend on the vendor implementation. A suggested approach will be added to the next release of this document. Please contact the Ontario Health Architecture Program office at oh-ds_architecture@ontariohealth.ca if there is an immediate need to implement this.

1. The client application starts writing to the CMS as soon as the user logs in with their ONE ID user account and gets an *access_token*.
2. The client application needs to use the *access_token* for all CMS calls.

3. The client application makes all calls to the CMS service through the Provider OAG and must include the request and response headers below, which are useful for logging and debugging:
    a. Request Headers
        i. *X-Request-Id* (a UUID generated by the application calling the Provider OAG)
        ii. *X-Gtwy-Client-Id*
        iii. *X-Gtwy-Client-Secret* (if you are a confidential client)
    b. Response Headers:
        i. *X-Correlation-Id*
        ii. *X-LobTxId*
4. When a user logs in for the first time, the client application first checks if the *$launch* ID has been assigned a value equal to the *hub.topic* value in the current CMS context session. If it does not, it makes a POST call to *$hub.url/createHubTopic* to create one and assigns it to the *$launch* ID. Sample messages and values are described in more detail in the CMS iGuide.

⚠ For different context sessions across multiple tabs, you must have a unique *$launch* ID for each tab.

9.3 CMS OH.userLogin Use Case

1. For all event calls to CMS, the client application must provide the following key-value pairs:
    - *id* = unique transaction ID; it is strongly advised to assign the *X-Request-Id* from the HTTP header as the value to this parameter
    - *timestamp* = current time as ISO 8601-2 timestamp in UTC
    - *event.['hub.Event']* = <event being triggered/submitted>
    - *event.['hub.Topic'] = $launch (hub.topic)*
2. Right after login, the client application sets the following values by submitting the *OH.userLogin* event to CMS. This is to set the context with the FHIR resource values as described here in the CMS iGuide. For example, for initial login:
    - *id* = unique transaction ID; it is strongly advised to assign the *X-Request-Id* from the HTTP header as the value to this parameter
    - *timestamp* = current time as ISO 8601-2 timestamp in UTC
    - *event.['hub.Event'] = OH.userLogin*
    - *event.['hub.Topic'] = $launch*
    - *event.context.filter(item => item.key == "organization")*
    - *event.context.filter(item => item.key == "location")* only if location is different than organization address, otherwise set to NULL
    - *event.context.filter(item => item.key == "practitioner")*

    - For the [Practitioner](#) FHIR resource, the *Practitioner.identifier.value* would be the practitioner's license number for the corresponding URI. If the user is not a practitioner that has a designated license of the supported URIs, then this is skipped
- *event.context.filter(item => item.key == "parameters").resource.parameters.filter(subItem => subItem.name == "appLanguage")*

10.0 Additional CMS Use Cases

10.1 CMS OH.Organization-change Use Case

Pre-requisite Event: *OH.userLogin*.

If the user is switching UAO after the Login sequence is executed, the client application must submit another authorization request through the ONE ID OAuth 2.0 flow, this time including *uao* as a parameter to the authorization request. Once the authorization flow is complete, the application must update the access token, the id token, and update the *inheritanceID* with the new *authzid*. The client application then submits an *OH.Organization-change* event to the CMS to update the *Organization* context and *Location* context (only if different from *Organization*) with the FHIR resource values as described here in the CMS iGuide.

⚠ If any clinical viewers or Viewlets are open and rely on context, the client application or end user must close and relaunch them when switching UAOs so that the context stays in sync.

10.2 CMS Patient-open Use Case

Pre-requisite Event: *OH.userLogin*.

When a user searches for and selects a patient, the client application must check if a patient is already open in the user session or not. If the user is switching between patients, then the client application must first issue a *Patient-close* event, followed by a *Patient-open* event. If the user opens up the patient for the first time, only a *Patient-open* event to the CMS is required to set the context with the FHIR resource values as described here in the CMS iGuide.

10.3 CMS OH.Patient-close Use Case

Pre-requisite Event: *Patient-open*.

If the user closes the patient in the client application, the application submits a *Patient-close* event to the CMS to remove the fully populated *Patient* context with the FHIR resource values as described here in the CMS iGuide.

11.0 Logout Use Case

11.1 CMS userLogout Use Case

Pre-requisite Event: *OH.userLogin*.

1. When the client application wants to close the current CMS session, because the application is closing or any other reason, it submits a *userLogout* event to the CMS to indicate the user has ended their session and to clear all the context and close the session. The usage and details of this event can be found here in the CMS iGuide.
2. Following the *userLogout* event, if the client application is closing, then it must invoke the ONE ID OAuth 2.0 logout URL as per OIDC specifications to end its ONE ID session.

12.0 Client Application Direct Integration Use Cases

12.1 Invoke PCOI Viewlet Use Case

Refer to Appendix B.

12.2 CMS OH.consentTargetChange Use Case

Refer to Appendix B.

12.3 Close Viewlet Use Case

Refer to Appendix B.

13.0 Error Handling

13.1 Client Application Errors

If the client application writing to CMS suddenly terminates or behaves abnormally, then it is suggested to go through the Login Use Cases again, create new tokens and a new CMS session with a new *hub.topic*. The lost context session is cleared from the CMS cache after 8 hours when the ONE ID session expires. If a Viewlet, clinical viewer or ClinicalConnect is open in any tabs, it is suggested that the user closes them before relaunching the client application so that it can get attached to the new context session that is created.

13.2 CMS Errors

Most errors are handled by the Viewlet and any errors that occur while launching the Viewlet should be handled by the browser's default behaviours. Since CMS is directly called by the client application, any CMS errors will need to be handled by the application. Errors returned from CMS will follow the structure of the example below. It lists the status, HTTP error code as code, and details of the initial call such as *hub.topic*, *hub.event*, *id* and *lob.id*. It then lists the *reason* for the error, with the *name*, a *message* and a list of CMS specific *error codes* in the table blow. The *errorCodes* is an array because it could have multiple issues.

In the example below, there is a *CMS_ERROR_INVALID_PUBLICATION* code because the data trying to be published is incorrect; then there is another code *CMS_ERROR_INVALID_HUBEVENT*, which is more in depth letting the caller know that the *hub.event* was invalid (*patient-ope* instead of *patient.open*).

To match the CMS error to the original call that was sent, the *X-Request-Id* in the client application's original CMS call should be the same as the *X-Correlation-Id* in the CMS error response.

```json
{
  "status": "Invalid",
  "code": 400,
  "id": "6b22f6a1-4e13-4043-9d2e-62622c78b7e7",
  "hub.topic": "D8584B7C5F3740BE960FBC20FA784613",
  "hub.event": "patient-ope",
  "lob.id": "6411817C08AC46B1AE72F605723F4D2C",
  "reason": {
    "name": "ValidationError",
    "message": "publication information is invalid",
    "errorCodes": [
      "CMS_ERROR_INVALID_PUBLICATION",
      "CMS_ERROR_INVALID_HUBEVENT"
    ],
    "hasErrorCodes": true
  },
  "success": false,
  "failure": true,
  "errorCodes": [
    "CMS_ERROR_INVALID_PUBLICATION",
    "CMS_ERROR_INVALID_HUBEVENT"
  ],
  "hasReason": true,
  "hasErrorCodes": true
}
```

13.3 HTTP Response Codes

Response Code	Description	Notes
200	OK	
204	No Content	
400	Bad Request	Resource could not be parsed or failed basic FHIR validation rules (or multiple matches were found for the same request)
404	Not Found	Resource type not supported, or not a FHIR endpoint
405	Method Not Allowed	The resource did not exist prior to the update, and the server does not allow client defined ids
500	Internal Server Error	

13.4 Response 400 Error Codes

Error Code	Description
CMS_ERROR_SYSTEM	Unexpected and untracked error
CMS_ERROR_CACHE	Cache operation error
CMS_ERROR_EVENT	Event handling error
CMS_ERROR_PARSE	JSON string parsing error
CMS_ERROR_STRINGIFY	JSON object stringify error

Error Code	Description
CMS_ERROR_SERIALIZE	TypeScript object to JSON object serialization error
CMS_ERROR_DESERIALIZE	JSON object to TypeScript object deserialization error
CMS_ERROR_CONTEXT_SYNC	Session context sync error
CMS_ERROR_REQUIRED_SESSION	Session context is required error
CMS_ERROR_REQUIRED_ORGANIZATION	Session context organization resource is required
CMS_ERROR_REQUIRED_PRACTITIONER	Session context practitioner resource is required
CMS_ERROR_INVALID_PRACTITIONER	Session context practitioner resource is invalid
CMS_ERROR_REQUIRED_PATIENT	Session context patient resource is required
CMS_ERROR_INVALID_PATIENT	Session context patient resource is invalid
CMS_ERROR_REQUIRED_PARAMETERS	Session context parameters resource is required
CMS_ERROR_REQUIRED_CONTEXT	FHIRcast publication request context array is required
CMS_ERROR_INVALID_CONTEXT	FHIRcast publication request context array is invalid
CMS_ERROR_REQUIRED_PUBLICATION	FHIRcast publication request is required
CMS_ERROR_INVALID_PUBLICATION	FHIRcast publication request is invalid
ERROR_REQUIRED_TIMESTAMP	FHIRcast publication request timestamp element is required
ERROR_REQUIRED_PUBLICATION_ID	FHIRcast publication request id element is required
ERROR_REQUIRED_EVENT	FHIRcast publication request event element is required
ERROR_REQUIRED_HUBTOPIC	FHIRcast publication request hubtopic element is required
CMS_ERROR_REQUIRED_HUBEVENT	FHIRcast publication request hubevent element is required
CMS_ERROR_INVALID_HUBEVENT	FHIRcast publication request hubevent element is invalid
CMS_ERROR_REQUIRED_CONTEXT_KEY	FHIRcast publication request context key element is required
CMS_ERROR_REQUIRED_CONTEXT_RESOURCE	FHIRcast publication request context resource element is required
CMS_ERROR_REQUIRED_EVENT_RESOURCE	FHIRcast event must have all required FHIR resources
CMS_ERROR_INVALID_EVENT_RESOURCE	FHIRcast event must have only allowed FHIR resources
CMS_ERROR_REQUIRED_EVENT_PARAMETER	FHIRcast event must have all required FHIR parameters
CMS_ERROR_INVALID_EVENT_PARAMETER	FHIRcast event must have only allowed FHIR parameters
CMS_ERROR_INVALID_RESOURCE	FHIR resource is invalid
CMS_ERROR_REQUIRED_HEADER	HTTP header is required
CMS_ERROR_INVALID_HEADER	HTTP header is invalid
CMS_ERROR_REQUIRED_SUB	Authorization header sub element is required
CMS_ERROR_REQUIRED_AZP	Authorization header azp element is required
CMS_ERROR_REQUIRED_UAO	Authorization header uao element is required
CMS_ERROR_REQUIRED_UAO_NAME	Authorization header uao name element is required
CMS_ERROR_REQUIRED_UAO_TYPE	Authorization header uao type element is required

Errors generated as the result of ONE ID OAuth 2.0 flows, LOB flows, or Provider OAG calls should be handled according to those specifications.

Appendix A: Viewlet Framework Response Codes

A.1 Introduction

Viewlets need to return a response code back to the calling application or may need to broadcast a response to multiple applications running in the same session. This appendix will cover current use cases as required by the PCOI Viewlet and will expand as more Viewlets are developed.

A.2 Events

For the initial implementation of the PCOI Viewlet, its responses will be based on events.

An event could be sent through a combination of different channels or through just one channel such as WebRTC, WebSockets, FHIRcast or web browser. For PCOI, the initial event handling is done in the web browser and uses the browser's event listeners and handlers through *window.postMessage*.

A.3 Response Messages

The Viewlet responds with an object containing three major sections: errors, successes and utility. Errors and successes are codes returned from a Viewlet's business logic and are also arrays as a single Viewlet can make calls to multiple lines of business (LOB) downstream.

It is possible one or more of the downstream calls can fail while one or more are successful. When an end user encounters a patient record with consent directives across multiple LOBs (e.g. DHDR and OLIS), they can choose to override the consent on all LOBs through the PCOI Viewlet interface. The PCOI Viewlet calls the PCOI microservice through the Provider ONE Access Gateway PCOI API and the microservice would in turn call each of the requested LOBs separately. Since two different LOBs were called, there is a possibility one could return a success, such as DHDR and the other a failure, such as OLIS. Each of the success/error objects will have the following fields:

- *code* - response code from the call.
- *microservice* - the LOB that this specific success or failure was for.
- *reason* - reason for the failure.
- *lobTxId* - key-value pairs of transaction IDs for all the downstream LOBs called by the microservice that can be used for troubleshooting.
- *utility* - specifies an array of codes to be sent by the Viewlet directly back to the parent application. For example, PCOI_CONSENT_CANCELLED, which requires the parent application to close the PCOI iFrame.

A.4 Message Format

The message will always be a JSON object with the following format:

```
{
    "appVersion": "<version Number>",
    "errors": [{ //(optional)
      "code": […],
      "microService": "name of the microservice or external API called",
      "reason": "response code and the short message",
      "lobTxId": {"lobkey":"txId", …} //contains all the LOB transaction IDs that can be used for troubleshooting.
    }],
    "successes": [{   //(optional)
      "code": […],
      "microService": "name of the microservice or external API called",
      "reason": "response code and the short message",
      "lobTxId": {"lobkey":"txId", …} //contains all the LOB transaction IDs that can be used for troubleshooting.
    }],
    "utility": {
      "code": […]
    }
}
```

A.5 Response Codes

A.5.1 Success Codes

The *PCOI_CONSENT_SUCCESS_01* success code confirms the call to the PCOI microservice completed successfully but does not confirm if the call to the DHDR LOB was successful.

The *201* and *PCOI_CONSENT_SUCCESS_02* consent codes confirm the call to the DHDR LOB was successful.

Code	Reason	Microservice
PCOI_CONSENT_SUCCESS_01	PCOI Transaction(s) Successful	PCOI
PCOI_CONSENT_SUCCESS_02	DHDR Transaction Successful	DHDR

A.5.2 Error Codes

Code	Reason	Microservice
PCOI_CORE_MISSING_HEADER_KEY	Missing HTTP Header	PCOI
PCOI_CORE_MISSING_HEADER_VALUE_01	Value must be 'application/fhir+json'	PCOI
PCOI_CORE_MISSING_BODY_VALUE	Invalid value: <VALUE>	PCOI
PCOI_CORE_VALIDATION_01	Token RID does not match Practitioner identifier	PCOI
PCOI_CORE_VALIDATION_03	Value does not match Enumerated list	PCOI
PCOI_CORE_UNHANDLED	Internal Server Error	PCOI
PCOI_CONSENT_VALIDATION_01	No valid override target specified	PCOI
PCOI_CONSENT_VALIDATION_02	Consent performer is not Patient or RelatedPerson	PCOI

Code	Reason	Microservice
PCOI_CONSENT_TIMEOUT	Downstream system did not return timely response	PCOI
PCOI_CONSENT_DHIR_OOS	DHIR override not available yet	DHIR
PCOI_CONSENT_UNHANDLED	Internal Server Error	PCOI
PCOI_CORE_MISSING_HEADER_VALUE_02	Value must be 'en' or 'fr'	PCOI

A.5.3 Utility Codes

Code	Reason	Microservice
PCOI_CONSENT_CANCELLED	N/A	N/A

A.5.4 Sample Message Code

```
{
    "appVersion": "1.0",
    "errors": [{
        "code": ["###", "###"],
        "microService": "PCOI",
        "reason": "501 DHIR Not In Service",
        "lobTxId": {"DHDR":"2ef028a9-645b-48f7-9c58-a39f2ce8fa1c","PCOI":"973c0760-d630-4059-8ccb-
13fa02e9356c"}
    },
    {
        "code": ["###", "###"],
        "microService": "PDF",
        "reason": "500 Internal Server Error",
        "lobTxId": {"DHDR":"2ef028a9-645b-48f7-9c58-a39f2ce8fa1c","PCOI":"973c0760-d630-4059-8ccb-
13fa02e9356c"}
    }],
     "successes": [
      {
        "code": ["PCOI_CONSENT_SUCCESS_02", "201"],
        "microService": "DHDR",
        "reason": "success",
        "lobTxId": {"DHDR":"2ef028a9-645b-48f7-9c58-a39f2ce8fa1c","PCOI":"973c0760-d630-4059-8ccb-
13fa02e9356c"}
      },
      {
        "code": ["PCOI_CONSENT_SUCCESS_01"],
        "microService": "PCOI",
        "reason": "success",
        "lobTxId": {"DHDR":"2ef028a9-645b-48f7-9c58-a39f2ce8fa1c","PCOI":"973c0760-d630-4059-8ccb-
13fa02e9356c"}
      }
    ],
    "utility":{
      "code": ["PCOI_CONSENT_CANCELLED"]
    }
}
```

Appendix B: PCOI Viewlet Integration

B.1 Introduction

The Provincial Consent Override Interface (PCOI) is a turnkey EHR consent override service offered by Ontario Health to trusted partners. Upon completion of one-time integration work, PoS applications will automatically gain access to UI improvements and additional LOB offerings as they are released by the project team.

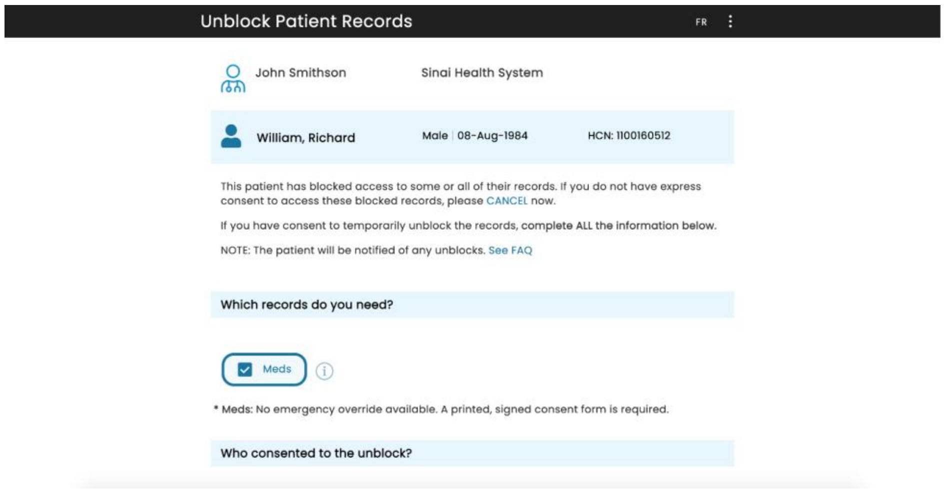
The first release of PCOI service supports the temporary override of consent directives on patient records in the Digital Health Drug Repository (DHDR). It is also ready to support Digital Health Immunization Repository (DHIR) consent overrides when the line of business implements it, which currently it has not.

The PCOI Viewlet is a single-page application (SPA) developed by Ontario Health, written in Angular® and served securely over HTTPS (using a public Entrust certificate) by scalable and containerized NGINX servers hosted on the agency’s on-premises Red Hat OpenShift platform. Other Ontario Health Viewlets currently in development and planned for future development are intended to follow this model.

B.2 PCOI Viewlet User Interface

The end user interactions with the PCOI Viewlet are briefly described here to help client application developers who are integrating their systems with it.

Step 1: Review Patient and Practitioner information and select the type of records which you would like to override.



John Smithson Sinai Health System

William, Richard Male | 08-Aug-1984 HCN: 1100160512

This patient has blocked access to some or all of their records. If you do not have express consent to access these blocked records, please CANCEL now.

If you have consent to temporarily unblock the records, complete ALL the information below.

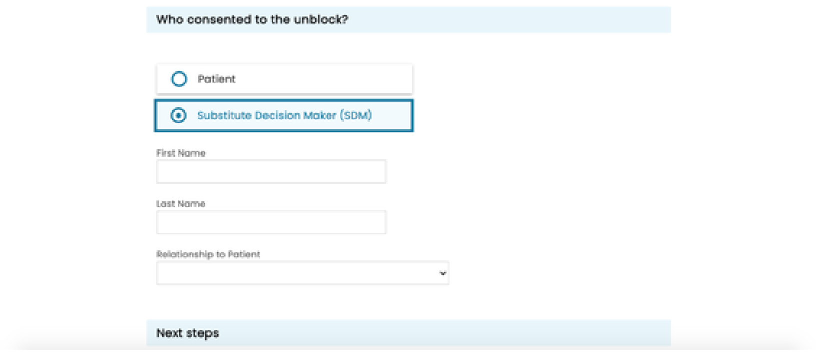
NOTE: The patient will be notified of any unblocks. See FAQ

Which records do you need?

* Meds: No emergency override available. A printed, signed consent form is required.

Who consented to the unblock?

Step 2: Select who is consenting to the unblock. If it is a Substitute Decision Maker (SDM), additional information will be required: first name, last name, and relationship to the patient.



Who consented to the unblock?

Patient

[✓ Substitute Decision Maker (SDM)]

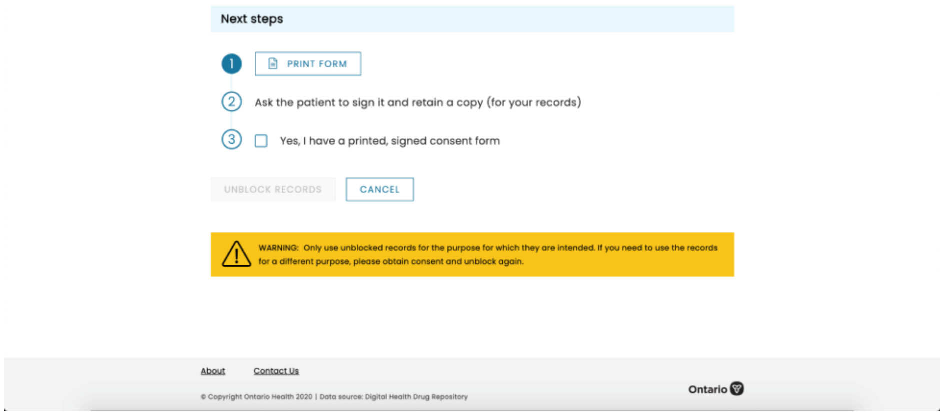
First Name

Last Name

Relationship to Patient

Next steps

Step 3: Review and follow the steps provided in the "Next Steps" section.



Next steps

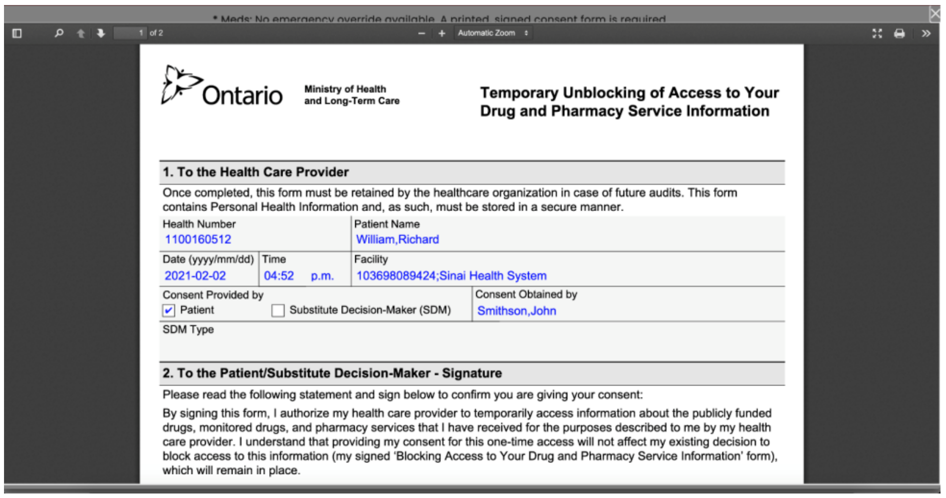
2 Ask the patient to sign it and retain a copy (for your records)

3 Yes, I have a printed, signed consent form

WARNING: Only use unblocked records for the purpose for which they are intended. If you need to use the records for a different purpose, please obtain consent and unblock again.

© Copyright Ontario Health 2020 | Data source: Digital Health Drug Repository

Step 4 (specific to the DHDR LOB): Print the form and have the patient sign the form.



* Meds: No emergency override available. A printed, signed consent form is required

Temporary Unblocking of Access to Your Drug and Pharmacy Service Information

1. To the Health Care Provider

Once completed, this form must be retained by the healthcare organization in case of future audits. This form contains Personal Health Information and, as such, must be stored in a secure manner.

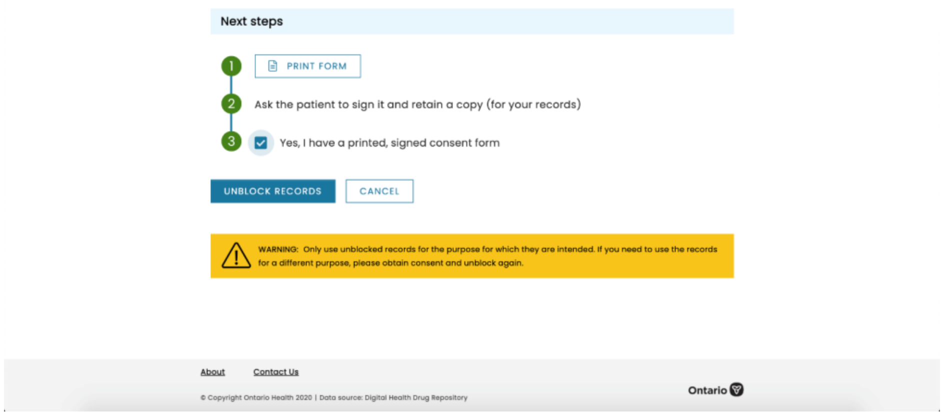
Health Number 1100160512		Patient Name William,Richard
Date (yyyy/mm/dd) 2021-02-02	Time 04:52 p.m.	Facility 103698089424;Sinai Health System
Consent Provided by [✓ Patient] Substitute Decision-Maker (SDM)		Consent Obtained by Smithson,John
SDM Type		

2. To the Patient/Substitute Decision-Maker - Signature

Please read the following statement and sign below to confirm you are giving your consent:

By signing this form, I authorize my health care provider to temporarily access information about the publicly funded drugs, monitored drugs, and pharmacy services that I have received for the purposes described to me by my health care provider. I understand that providing my consent for this one-time access will not affect my existing decision to block access to this information (my signed 'Blocking Access to Your Drug and Pharmacy Service Information' form), which will remain in place.

Step 5: Verify the consent form is printed and signed then click the checkbox.


Next steps

PRINT FORM

Ask the patient to sign it and retain a copy (for your records)

Yes, I have a printed, signed consent form

UNBLOCK RECORDS CANCEL

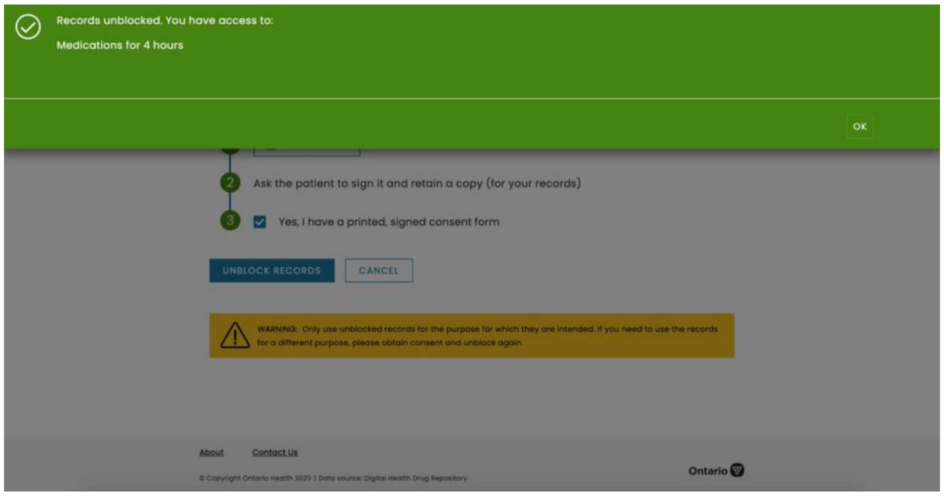
WARNING: Only use unblocked records for the purpose for which they are intended. If you need to use the records for a different purpose, please obtain consent and unblock again.

About Contact Us

© Copyright Ontario Health 2020 | Data source: Digital Health Drug Repository

Ontario

Step 6: Click the *Unblock Records* button and receive the success message back from the PCOI service.


Records unblocked. You have access to:

Medications for 4 hours

OK

Ask the patient to sign it and retain a copy (for your records)

Yes, I have a printed, signed consent form

UNBLOCK RECORDS CANCEL

WARNING: Only use unblocked records for the purpose for which they are intended. If you need to use the records for a different purpose, please obtain consent and unblock again.

About Contact Us

© Copyright Ontario Health 2020 | Data source: Digital Health Drug Repository

Ontario

It is recommended to follow the Conceptual Flow Diagram and Use Cases Sequence Diagram when reading about the client application interactions with the PCOI Viewlet described in this appendix.

B.3 User Encountering DHDR Consent Block

1. The client application queries the DHDR FHIR API on the Provider Gateway with the ONE ID OAuth 2.0 token. The DHDR response returns no records due to the presence of an active consent directive identified by the specific *severity* and *code* key-value pairs shown in the sample JSON object below. The client application stores the initial DHDR query to resubmit later when the consent override flow completes.

```
{
  "severity": "warning",
  "code": "suppressed",
  "details": {
    "text": "Some information is blocked. Additional information may be
      available using the Temporary Consent Unblock Access Protocol"
  }
}
```

2. Once confirmed that there is a consent block on the DHDR results, the client application would display a message to the end user along with an override button that they would click on to launch the PCOI Viewlet.

B.4 Invoke PCOI Viewlet

Prerequisite: Some applications will choose not to write CMS events as they occur, but rather just-in-time (JIT) before invoking the PCOI Viewlet. In this case, the client application must ensure all Login Use Cases (authorization, createHubTopic, OH.userLogin) and Patient-open events are completed first before initiating the following steps:

⚠ For client applications that support switching UAOs within the same login session: if the PCOI Viewlet is open, the client application must close and relaunch it after the UAO switch to ensure context stays in sync.

B.4.1 OH.consentTargetChange Use Case

1. When the user clicks on the override button, the client application submits an *OH.consentTargetChange* event to the CMS. This is to set the context with the CMS to indicate which LOB(s) have been triggered for an override. Consider the following when developing:
   - The first release of PCOI only supports the DHDR LOB.
   - When starting the consent override invocation process, send all EHR-sourced consent flags to the CMS.
   - As additional LOBs are added to future releases of PCOI, the client application must be able to handle the list of LOBs dynamically. This is to avoid additional code changes in the future when new LOBs are added.
   - Further details on the usage of this event can be found here in the CMS iGuide.
2. The client application code will launch the PCOI URL, which was previously parsed from toolbar and saved as *$PCOI_URL*, in an iFrame in the following matter:
   - The client application will create an iFrame as a modal window. It is recommended to use focus/modal so the user can only interact with this window and not click outside of it.
   - The iFrame will be opened with restrictions by adding the *sandbox* attribute to it with the following values: *allow-forms*, *allow-scripts*, *allow-same-origin* and *allow-modals.*
   - As shown in **Step 4b** of the Conceptual Flow Diagram, the client application will launch the PCOI Viewlet with the proper URL, which includes:
     - *hub.topic* as a query parameter named *launch*;
     - *FHIR_iss* as a query parameter named *iss*; and
     - *authzid* as a query parameter named *inheritanceID.*
     - Sample code:

```
<iframe id="pcoi-frame" [src]="https://pcoi.ecp.ehealthontario.ca
   ?launch=…&iss=…&inheritanceID=…" sandbox="allow-forms allow-scripts allow-same-origin
allow-modals" ></iframe>
```

- The client application will have a window listener implemented to listen to a *window.postMessage* response from the iFrame it just launched. The message will come back in the Viewlet Framework defined response format as discussed in Appendix A.

B.4.2 User Interacts with the PCOI Viewlet

1. The PCOI Viewlet takes advantage of single sign-on (SSO) and the browser session. This allows the PCOI Viewlet to pick up and re-use the login session (authentication). It can also make its own authorization calls and process the resulting ONE ID OAuth 2.0 *access_token*.
2. The PCOI Viewlet makes use of the CMS to get the context of the current user session. The resources that come back are all FHIR resources and FHIR extensions. The CMS *GET* call syntax, usage and FHIR resource values are described in more detail in the CMS iGuide.
3. The end user interacting with the PCOI Viewlet is asked to: answer specific questions regarding the consent override request, print the form and submit the request. Upon submitting, the Viewlet makes calls to the PCOI FHIR API and processes the request against every LOB that was indicated.
4. Once the override is complete, the PCOI Viewlet will display a message to the end user with the results. As more LOBs are added to the PCOI service, results may indicate that an override succeeded for some while failed for others depending on the downstream responses from each LOB. Upon dismissing the message and closing the PCOI Viewlet, the Viewlet sends a notification message to the parent window (through *window.postMessage*) with the result in JSON format as outlined in Appendix A.
5. If there is any other reason the iFrame closes and there is no message received from the PCOI Viewlet, then assume a serious error occurred and the consent override did not go through. Approach this as a cancelled transaction and restart the flow with the initial LOB call that triggered consent override. It is possible that the consent override processed successfully, and the unexpected error came after the fact, so in this case, the consent override would not be triggered. Therefore, it is advised the client application attempt to re-load the LOB data and if the record still has a consent block, then prompt the user to override again.
6. If the user manages to click outside of the PCOI Viewlet window, the client application should not allow any further user interactions until the PCOI Viewlet closes.

B.5 User Interactions and Client Application Behaviour after Consent Override

For the consent override requests that completed successfully, the LOB query that initially triggered the consent block message(s) should be re-submitted. It is up to the client application to decide if the re-submission will happen automatically or if the user should be prompted to re-submit the query. After re-submission of the LOB query, a full response message would be returned with the clinical details. The returned FHIR resource includes an additional attribute that would indicate that this response has been provided because of a consent override. This should be returned by the client application to the end user for their awareness.

A sample response from DHDR:

```
{
  "severity": "information",
  "code": "informational",
  "details": {
     "coding": [
       {
         "code": "CONSENT_TEMP_UNBLOCK"
       }
     ],
     "text": "Patient has temporarily unblocked access to view and
       use drug information."
     }
}
```

Appendix C: ClinicalConnect^{TM} and Clinical Viewer Integration

C.1 Introduction

Some of today's EMRs and HISs can launch a provincial clinical viewer such as ClinicalConnect^{TM} or ConnectingOntario through one click of a button. This is achieved by using ONE ID single sign-on (SSO) and some mechanism for patient context. A clinician using a local EMR can easily access one of these clinical viewers from within the application to retrieve a broader set of data from the patient's EHR assisting them in clinical decisions and in turn improving patient outcomes.

The current release of the Viewlet Framework supports integration and one-click launch of ClinicalConnect and other clinical viewers using ONE ID OAuth 2.0 for SSO and Ontario Health's Context Management System (CMS) for patient context.

A future release of Viewlet Framework and CMS is expected to additionally support ConnectingOntario and other clinical viewers that require the use of WebRTC, WebSockets, webhooks and postback/callback URLs as outlined in the FHIRcast specification.

C.2 Invoking ClinicalConnect or a Clinical Viewer

Pre-requisite: Some applications will choose not to write CMS events as they occur, but rather just-in-time (JIT) before invoking ClinicalConnect or another clinical viewer. In this case, the client application must ensure all Login Use Cases (authorization, createHubTopic, OH.userLogin) and Patient-open events are completed first before initiating the use case below.

It is recommended to follow the Conceptual Flow Diagram and Use Cases Sequence Diagram when reading about the client application interactions with ClinicalConnect or another clinical viewer as described in this appendix.

C.3 Invoke ClinicalConnect or a Clinical Viewer Use Case

For client applications that support switching UAOs within the same login session: if ClinicalConnect or the clinical viewer is open, the client application or end user must close and relaunch it after the UAO switch to ensure context stays in sync.

As shown in **Step 4b** of the Conceptual Flow Diagram, the client application will launch ClinicalConnect or another clinical viewer with the proper URL, which includes:

- *hub.topic* as a query parameter named *launch*
- *FHIR_iss* as a query parameter named *iss*
- *authzid* as a query parameter named *inheritanceID*

- *ONE ID Identity Provider URL, https://federation.ehealthontario.ca/fed/idp*, as a query parameter named *SelectedIDP*

C.4 Launch URL Format

```
https://my.viewer.ca?launch=...&iss=...&inheritanceID=...&SelectedIDP=https://federation.ehealthontario.ca/fed/idp
```