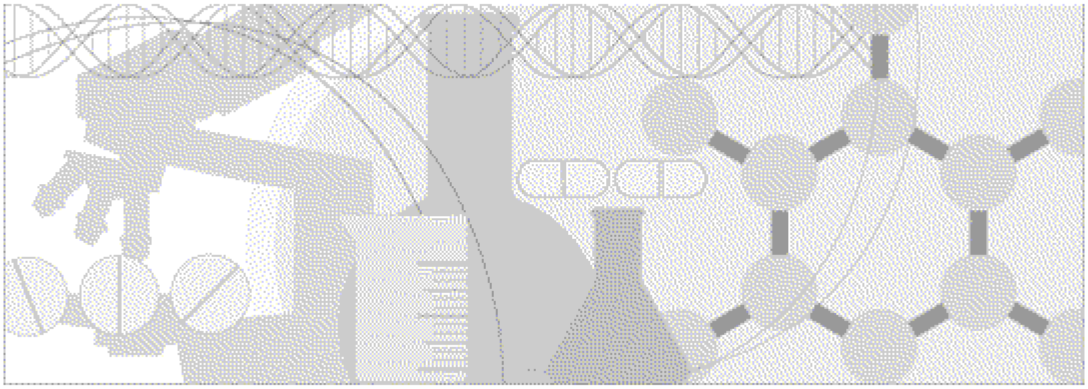
Secure Development Guide

Oracle® Health Sciences Central Designer
Release 2.1.2



Part Number: E72241-01

Contents

About this guide

In this preface

Overview of this guide

The *Secure Development Guide* provides an overview of common security risks for developers using Application Programming Interfaces (APIs) with the Oracle® Health Sciences Central Designer application, and information on how to address those risks.

Note: The set of recommendations in this document is not exhaustive and Oracle provides no guarantee that implementing all the suggestions in this document provides sufficient protection for all security threats. The reason for this disclaimer is that you cannot delegate responsibility for secure application development to a third party or a single document. The purpose of this document is to help developers know what security tools and features they can use to implement application security. This document does not replace a formal code review process.

Audience

This guide is for developers using Application Programming Interfaces (APIs) with the Oracle® Health Sciences Central Designer application.

Documentation

The product documentation is available from the following locations:

- **My Oracle Support** (https://support.oracle.com)—*Release Notes* and *Known Issues*.
- **Oracle Technology Network** (http://www.oracle.com/technetwork/documentation/hsgbu-154445.html)—The most current documentation set, excluding the *Release Notes* and *Known Issues*.

If the software is available for download, the complete documentation set is available from the Oracle Software Delivery Cloud (https://edelivery.oracle.com).

All documents may not be updated for every Central Designer release. Therefore, the version numbers for the documents in a release may differ.

Item	Description
Release Notes	The *Release Notes* document provides detailed information about the requirements, enhancements, and fixed issues in the current release.
Known Issues	The *Known Issues* document provides detailed information about the known issues in this release, along with workarounds, if available.
Installation Guide	The *Installation Guide* provides system requirements and instructions for installing and upgrading the Oracle® Health Sciences Central Designer software and the Oracle® Health Sciences Central Designer Administrator software.
Administrator Guide	The *Administrator Guide* describes how to use the Oracle® Health Sciences Central Designer Administrator software to set up users, permissions, system configuration parameters, and catalog defaults.
User Guide	The *User Guide* introduces the study design environment in the Oracle® Health Sciences Central Designer application and describes how to work as a study design team in that environment, including how to: • Work collaboratively. • Maximize study design efficiency by reusing study objects. • Manage collections of study objects.
InForm Design Guide	The *InForm Design Guide* describes how to design a study for deployment to the InForm application.
Rules Reference Guide	The *Rules Reference Guide* is a reference to the tools that are available for creating rule expressions, including: • Study object properties. • Functions. • Constants. • Data mappings. • Methods, operators, and literals.

Item	Description
Secure Configuration Guide	The *Secure Configuration Guide* provides an overview of the security features provided with the Oracle® Health Sciences Central Designer application, including details about the general principles of application security, and how to install, configure, and use the Central Designer application securely.
Third Party Licenses and Notices	The *Third Party Licenses and Notices* document includes licenses and notices for third party technology that may be included with the Central Designer software.
Secure Development Guide	The *Secure Development Guide* provides an overview of common security risks for developers using Application Programming Interfaces (APIs) with the Oracle® Health Sciences Central Designer application, and information on how to address those risks.

Documentation accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

If you need assistance

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Finding Central Designer information and patches on My Oracle Support

The latest information about the Central Designer application is on the Oracle Support self-service website, My Oracle Support. Before you install and use the Central Designer application, check My Oracle Support for the latest information, including *Release Notes* and *Known Issues*, alerts, white papers, bulletins, and patches.

Creating a My Oracle Support account

You must register at My Oracle Support to obtain a user name and password before you can enter the site.

1 Open a browser to https://support.oracle.com.

2 Click the **Register** link.

3 Follow the instructions on the registration page.

Finding information and articles

1 Sign in to My Oracle Support at https://support.oracle.com.

2 If you know the ID number of the article you need, enter the number in the text box at the top right of any page, and then click the magnifying glass icon or press **Enter**.

3 To search the knowledge base, click the **Knowledge** tab, and then use the options on the page to search by:

- Product name or family.
- Keywords or exact terms.

Finding patches

You can search for patches by patch ID or number, product, or family.

1 Sign in to My Oracle Support at https://support.oracle.com.

2 Click the **Patches & Updates** tab.

3 Enter your search criteria and click **Search**.

4 Click the patch ID number.

The system displays details about the patch. You can view the Read Me file before downloading the patch.

5 Click **Download**, and then follow the instructions on the screen to download, save, and install the patch files.

Finding Oracle documentation

The Oracle website contains links to Oracle user and reference documentation. You can view or download a single document or an entire product library.

Finding Oracle Health Sciences documentation

For Oracle Health Sciences applications, go to the Oracle Health Sciences Documentation page at http://www.oracle.com/technetwork/documentation/hsgbu-clinical-407519.html.

Note: Always check the Oracle Health Sciences Documentation page to ensure you have the most up-to-date documentation.

Finding other Oracle documentation

1 Do one of the following:

- Go to http://www.oracle.com/technology/documentation/index.html.
- Go to http://www.oracle.com, point to the **Support** tab, and then click **Product Documentation**.

2 Scroll to the product you need, and click the link.

Finding prerequisite software for Oracle Health Sciences applications

Prerequisite software for Oracle Health Sciences applications is available from the following locations:

- Download the latest major or minor release from the Oracle Software Delivery Cloud (https://edelivery.oracle.com/).

  For information on the credentials that are required for authorized downloads, click **FAQ** on the main page of the Oracle Software Delivery Cloud portal.

- Download subsequent patch sets and patches from My Oracle Support (https://support.oracle.com).

  To find patch sets or patches, select the **Patches & Updates** tab.

If a previous version of prerequisite software is no longer available on the Oracle Software Delivery Cloud, log a software media request Service Request (SR). Previous versions of prerequisite software are archived and can usually be downloaded. After you open an SR, you can check its status:

- US customers: Call 1-800-223-1711.
- Outside the US: Check www.oracle.com/us/support/contact/index.html for your local Oracle Support phone number.

For more information on logging a media request SR, go to My Oracle Support for Document 1071023.1: Requesting Physical Shipment or Download URL for Software Media (https://support.oracle.com/epmos/faces/DocumentDisplay?id=1071023.1).

CHAPTER 1

Secure development overview

In this chapter

API overview

The Central Designer application provides an API for creating .NET DLLs to execute custom code (user-defined functions) called by rules through the Central Designer rule engine.

Rules and user-defined functions can be executed in the following places:

- On the Central Designer application client during interactive rule testing.
- On the Central Designer application server during study validation that executes rule tests.
- On the InForm application server during rule execution.

Because any .NET API can be called from these DLLs (file access, database access, and so on), developers must follow secure guidelines while developing the code. In addition, the Central Designer application provides a certificate-based mechanism to prevent untrusted DLLs from executing code that requires elevated permissions.

For more information on the .NET Code Access Security (CAS) model, see https://msdn.microsoft.com/en-us/library/dd233102(v=vs.100).aspx.

For secure coding guidelines for .NET applications, see https://msdn.microsoft.com/en-us/library/d55zzx87(v=vs.90).aspx.

Rule sandbox details

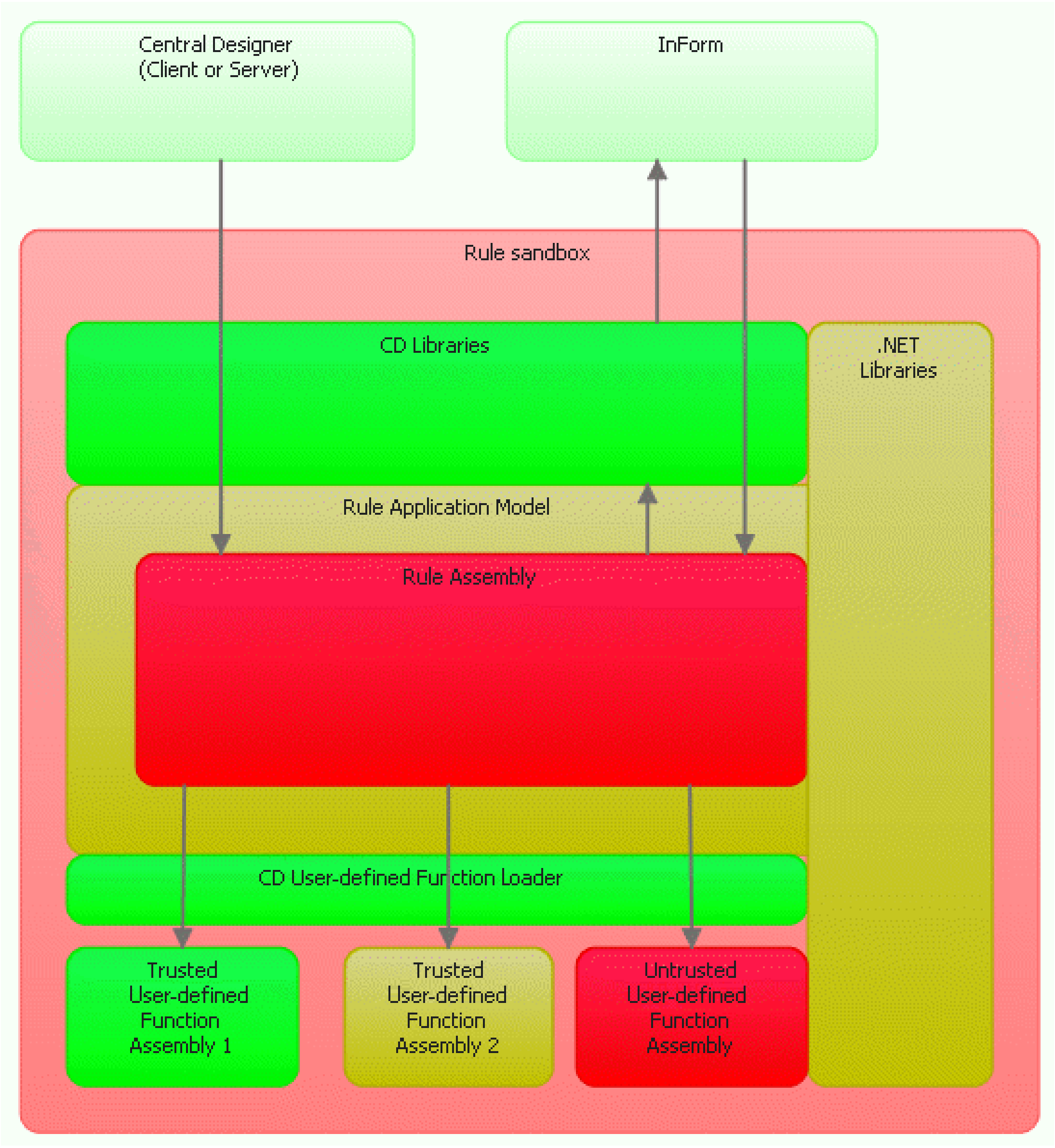
All user-defined function DLLs executed by the Central Designer rule engine run in a sandbox. The rule sandbox is a .NET AppDomain. You can think of it as a light-weight process inside the OS process.

The rule sandbox has the minimum set of permissions possible. This means the permission to execute code (SecurityPermissionFlag.Execution), which is the maximum restriction possible in .NET.

There are two types of user-defined function assemblies:

- **Trusted**—When loaded into the sandbox, code in trusted assemblies executes as safe-critical code or as security-critical, and can elevate permissions to perform extra operations, such as DB access.
- **Untrusted**—Untrusted user-defined function assemblies have the same permissions as a rule: only basic operations, which are not outlined in the *Rules Reference Guide* as requiring special permission. Untrusted code or transparent code can access only safe-critical code in assemblies marked with the attribute AllowPartiallyTrustedCallers.

The following diagram shows the relationship between the sandbox and the main application (InForm or Central Designer), and the assembly structure inside the sandbox.


Central Designer (Client or Server)
InForm
Rule sandbox
CD Libraries
.NET Libraries
Rule Application Model
Rule Assembly
CD User-defined Function Loader
Trusted User-defined Function Assembly 1
Trusted User-defined Function Assembly 2
Untrusted User-defined Function Assembly

Color codes:

- **Green**—Good, trusted code that does not allow partially trusted callers (security-critical).
- **Yellow**—Trusted code that can be called by untrusted code (security safe-critical).
- **Red**—Untrusted code (transparent code).

The two light-green boxes on top represent AppDomains for the main application: Central Designer or InForm. These domains run in full trust. AppDomain technology isolates them from the sandbox. Communication between AppDomains (represented by arrows on the diagrams) is done with remoting calls (inter-process communication or RPCs).

If you follow a call from the Central Designer box to invoke the rule from the rule assembly:

1 The call is first created in the sandbox.

2 The call loads Central Designer libraries inside the sandbox and uses a proxy to invoke the method from Central Designer libraries (calls a green code).

3 The Central Designer libraries load the rule assembly (red code) and invoke the method from the rule assembly.

4 The rule assembly can make calls through the Rule Application Model into Central Designer libraries (arrow going up).

5 The rule assembly can make calls through the Rule Application Model into user-defined functions (arrows going down).

> **Note: The code executed when any of these calls are made is restricted to the set of sandbox permissions. It does not matter if it is trusted or untrusted code, unless trusted code chooses to temporarily elevate the permissions. .NET does not propagate elevated permissions to untrusted code.**

Trusted assemblies

The rule sandbox has the following classes of trusted assemblies:

- Central Designer assemblies built by Oracle and signed by the corporate code signing tool are trusted (security-critical and security safe-critical).
- User-defined function assemblies with their public keys registered in the machine certificate store are loaded in the sandbox as yellow or green.

> **Note: The decision to trust these assemblies is based on a manual procedure of installing certificates in the certificate store. A manual procedure creates the possibility of human error by registering incorrect public keys and establishing trust for a bad assembly.**

While trusted assemblies run in the highly restrictive sandbox, they can elevate permissions. For the procedure and guidelines for such elevations, see ***Elevating CAS permissions*** (on page 6).

The only high-level restriction that you can specify at the assembly level is the SecurityTransparentAttribute, which prohibits all elevations in the assembly. If the assembly needs to elevate permissions, the only guards against over-elevating are code review and code analysis tools.

Untrusted assemblies

Untrusted assemblies consist of:

- One rule assembly generated by the Central Designer application from the study structure. The generated rule assembly includes rules written by users.
- User-defined function assemblies that do not come from Oracle Services. Such assemblies either do not have strong names or their public keys are not installed in the machine certificate store.

Oracle recommends that you use strong names for all user-defined function assemblies because it has many benefits besides establishing trust. The Central Designer application also supports existing assemblies that are not strongly named. The .NET application does not allow assemblies without strong names as trusted assemblies.

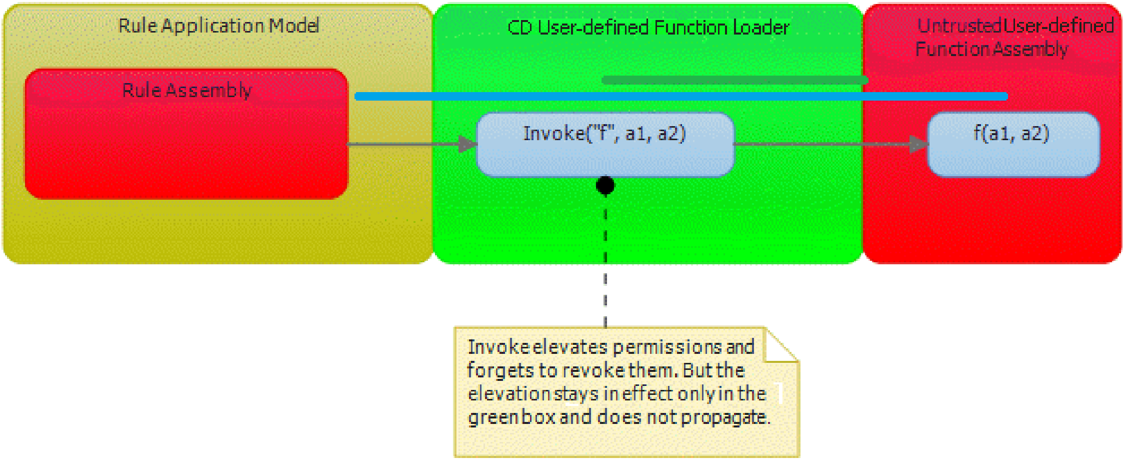
Elevating CAS permissions

Every elevated permission is a potential security hole. Untrusted code can use it to do something it cannot do directly (for example, by calling the .NET library directly). Ideally, there should be no permission elevations at all. However, developers may need access to restricted resources, such as config files, registry values, and calls using reflection. Access to restricted resources requires elevated permissions.

General principles

This diagram shows the permission flow. The blue line represents a basic set of permissions defined by the sandbox. The green line represents additional permissions added by the Invoke function.

The elevated permissions live only within the green box.


Rule Application Model
Rule Assembly
CD User-defined Function Loader
Invoke("f", a1, a2)
Untrusted User-defined Function Assembly
f(a1, a2)

Invoke elevates permissions and forgets to revoke them. But the elevation stays in effect only in the green box and does not propagate.

Guidelines for elevating CAS permissions

- Elevate permissions at the lowest level in the code, which is the closest point in the call stack that fails if the permission is not elevated.
- Elevate permissions only in the code specific to rule execution. If elevation is required at a lower level, move up in the call stack to the nearest call that is specific to rules.
- Elevate permissions on methods, and perhaps on classes, but not on the whole assembly.
- Be restrictive and specific. For example, if the code needs to read one registry key, elevate permission only for this key, not the whole registry.
- Use .NET attributes, not procedural code to elevate permissions. .NET attributes limit the scope of elevation to one method call or one class. In procedural code, you can forget to reset elevated permissions, and they can stay on. However, this sets a limit on how restrictive and specific you can be when you elevate permissions on methods or classes. For example, to limit the IO permission to one file, you must specify the complete path, which is known only at runtime, so this is impossible to do with an attribute.
- Add comments to each elevation explaining why it was necessary, and list the use case, unit test, or code path that requires that permission.

Preventing hijacking of elevated permissions

- Use the .NET code transparency model. Keep the amount of Central Designer code that untrusted code can call to the minimum. In the Central Designer application, very few assemblies allow partially untrusted callers.
- Elevate permissions only when necessary and only in the code that is rule specific.
- Review the possible code path from untrusted code to the function that elevates permission to see if untrusted code can misuse the function.
- During the review process, watch for too generic or too powerful functions running with elevated permissions.
  - One example of such a function is RunExternalFunction(assemblyName, className, methodName, parameters). It allows untrusted code to execute ANYTHING, and it is executed in full trust.

Preventing access to code outside the sandbox

One of the security exploits listed on the web is the ability of untrusted code to call Assembly.Load(assemblyFullName) to gain access to the code in a different assembly. To address this risk, keep untrusted assemblies in a folder separate from the bin folder of the hosting application. Oracle does this in all rule execution environments.

Preventing untrusted code from leaving the sandbox

See http://www.contextis.com/files/are_you_my_type.pdf, which explains several exploits based on .NET serialization, where an untrusted object can be serialized inside the sandbox, cross AppDomain boundaries, and be de-serialized already in the fully trusted AppDomain.

The rule sandbox does not have a permanent, must-use communication channel with the parent AppDomain, but several things might involve serialization from the sandbox to the parent AppDomain:

- Exceptions thrown from rule execution.
- Results of rule tests returned from the Rule Test Engine, including exceptions if the test failed.
- Calls from a rule into the InForm application through the COM interface.

Sandbox exceptions are caught inside the sandbox, and new exceptions are thrown to the parent AppDomain. The assumption is that all caught exceptions are already logged, and creating new exceptions will not result in the loss of information about errors.

If a test fails and an exception occurs, the exception is rewritten or replaced with a string error message. The Rule Test Harness uses only an exception message.

The InForm COM interface comes into the main fully trusted AppDomain as .NET COM Interop. This reference then passes into the sandbox. Several methods in this interface have object type parameters, which potentially can allow any object to pass in. Untrusted code can create a serializeable object, and pass it into the InForm interface. The object can be de-serialized into the main fully trusted AppDomain.

However, InForm Interop assemblies do not allow partially trusted callers, so untrusted code cannot call them directly. There is no way to pass indirectly arbitrary objects into these calls, because the InForm application assumes they are integers.

Limitations of the rule sandbox security model

The rule sandbox security model has several limitations:

- The following DoS attacks can occur:
  - Infinite sleep keeping the worker thread locked. Over time, such rules silently take all threads and starve the application.
  - Infinite loop consuming CPU cycles and slowing down the machine.
  - Writing a large amount of data to the event log and slowing down the application.
  - Allocating a large amount of memory.
- Clinical data corruption can use the legitimate interface of the Rule Application Model.
- Due to human error, the incorrect public key can be installed into the certificate store and establish trust for a bad user-defined function assembly.

Oracle Cloud for Industry (OCI) hosted Central Designer application

The Oracle Services team is responsible for reviewing the source code for any user-defined function DLL that requires elevated permissions and that will be used by studies hosted by OCI, even if the Central Designer instance is hosted by the customer.

The customer sends the source code to the Oracle Services team, who reviews and approves it, based on the security and coding standards maintained by Oracle.

The Oracle Services team then compiles the DLL and signs it using the Oracle signing tool (the same tool used to sign the Central Designer binaries). The signed DLL is then provided to the customer.

SQL injections

User-defined function DLLs can call any SQL statements that developers write, so you must adhere to secure SQL coding practices when writing SQL to be used by the user-defined function code.

XML injection

User-defined function DLLs consume any XML that developers need to use, so you must adhere to secure XML coding practices when manipulating XML in the user-defined function code.