

VERACODE

APIs

March 2020

Table of Contents

Introduction to the Veracode APIs.....	5
Getting Started	5
API Usage and Access Guidelines.....	5
Understanding API Access.....	6
API Roles.....	6
Generating API ID and Key Credentials.....	10
Generate Veracode API Credentials.....	10
Revoke Veracode API Credentials.....	11
Configuring the API Credentials File.....	11
Configure the API Credentials File on Windows.....	11
Configure the API Credentials File on macOS and Linux.....	12
Configure the API Credentials as Environment Variables on macOS and Linux.....	13
Enabling HMAC Authentication.....	13
Install the Java Authentication Library.....	15
Install the Python Authentication Library.....	15
Using HTTPie with the Python Authentication Library.....	16
HMAC Signing Example in Python.....	17
HMAC Signing Example in Java.....	17
HMAC Signing Example in C#.....	21
Veracode XML API Quick Reference Guide.....	25
Frequently Asked Questions (FAQ).....	34
Troubleshooting.....	35
 REST APIs.....	 42
Using the Veracode REST APIs.....	42
AppSec APIs.....	42
Applications API.....	42
Policy API.....	42
Manual Testing API.....	43
Use Case Scenarios.....	43
Dynamic Analysis APIs.....	44
Dynamic Analysis API.....	44
Use Case Scenarios.....	45
eLearning APIs.....	56
Using the Veracode eLearning APIs.....	56
Greenlight APIs.....	67
Greenlight API.....	67
Software Composition Analysis APIs.....	67
Veracode SCA Agent API.....	67
 XML APIs.....	 68
Using the Veracode XML APIs.....	68
Upload APIs.....	68
Using the Upload API.....	68
Upload API Calls.....	69

Mapping Upload API Tasks.....	70
beginprescan.do.....	70
beginscan.do.....	72
createapp.do.....	76
createbuild.do.....	80
deleteapp.do.....	81
deletebuild.do.....	82
getappinfo.do.....	83
getapplist.do.....	85
getbuildinfo.do.....	87
getbuildlist.do.....	89
getfilelist.do.....	91
getpolicylist.do.....	92
getprescanresults.do.....	94
getvendorlist.do.....	96
removefile.do.....	97
updateapp.do.....	98
updatebuild.do.....	102
uploadfile.do.....	103
uploadlargefile.do.....	105
API Prescan Status Information.....	107
API Build Status Information.....	108
API Tutorial: How to Scan an Application.....	109
Results APIs.....	111
Using the Results API.....	111
Results API Calls.....	111
Mapping Results API Tasks.....	112
detailedreport.do.....	112
detailedreportpdf.do.....	117
getaccountcustomfieldlist.do.....	118
getappbuilds.do.....	119
getcallstacks.do.....	122
summaryreport.do.....	124
summaryreportpdf.do.....	127
thirdpartyreportpdf.do.....	129
API Tutorial: How to Access Scan Results.....	130
Admin APIs.....	131
Using the Admin API.....	131
Admin API Calls.....	131
createuser.do.....	132
getuserinfo.do.....	136
getuserlist.do.....	137
updateuser.do.....	139
createteam.do.....	142
deleteteam.do.....	143
getteaminfo.do.....	144
getteamlist.do.....	146
updateteam.do.....	147
getcurriculumlist.do.....	148
gettracklist.do.....	149
getmaintenancescheduleinfo.do.....	150
Mitigation APIs.....	151
Using the Mitigation and Comments API.....	151
Mitigation and Comments API Calls.....	152

getmitigationinfo.do.....	152
updatemitigationinfo.do.....	154
API Tutorial: How to Use the Mitigation Calls.....	156
DynamicDS APIs.....	157
Using the DynamicDS API Calls.....	157
createdynamicscan.do.....	157
dynamicincludeexclude.do.....	159
submitdynamicscan.do.....	160
addbrowserbasedlogin.do.....	162
uploadformbasedloginscript.do.....	164
getdynamicstatus.do.....	165
getdynamicflaws.do.....	166
getvsalist.do.....	169
assignvsa.do.....	170
Flaw Report APIs.....	171
Using the Flaw Report API.....	171
generateflawreport.do.....	172
downloadflawreport.do.....	173
Sandbox APIs.....	176
Using the Sandbox APIs.....	176
createsandbox.do.....	177
getsandboxlist.do.....	178
promotesandbox.do.....	179
updatesandbox.do.....	180
deletesandbox.do.....	182
VAST APIs.....	183
Using the VAST APIs.....	183
sharedreport.do.....	183
sharedreportpdf.do.....	186
getsharedreportlist.do.....	187
getsharedreportinfo.do.....	188

API Wrappers..... 190

Using API Wrappers.....	190
Install and Use the C# API Wrapper.....	190
Install and Use the Java API Wrapper.....	191
Verify the Authenticity of Java Artifacts.....	192
Configure the Java Wrapper Parameters.....	192
Using the Veracode API Wrappers from the Command Line.....	193
Run the API Wrapper from the Command Line.....	194
API Wrapper Parameters.....	194
Simple Actions for the API Wrappers.....	196
Composite Actions for the API Wrappers.....	198
Using the API Wrappers as a Library or a Command-Line Application.....	201
Referencing the Veracode C# API Wrapper from Visual Studio.....	202
Referencing the Veracode Java API Wrapper from Eclipse.....	202
Have a Question??.....	203
Engage with the Veracode Community.....	203
Veracode Blog.....	204
Copyright 2020 VERACODE, All Rights Reserved.....	204

Introduction to the Veracode APIs

Getting Started

The Veracode REST and XML APIs mirror the major steps you complete on the Veracode Platform, automating the scanning, reviewing, mitigating, and administrative tasks. To understand how the APIs work, it is recommended that you familiarize yourself with the Veracode Platform workflow for scanning applications before you use the APIs.

The intended users of the Veracode APIs are developers who are members of the software development team responsible for performing the security checks on the software code. The objective of the APIs is to enable developers who work in rapid build and test cycles to fully automate security verification for entire software portfolios, and to integrate with internal build and bug-tracking systems. Instead of manually using the platform to go through the individual steps of configuring and submitting a scan request then reviewing the results, you can integrate the API calls directly into your IDE and build system code to scan early and often.



NOTE: Before using the XML APIs, [ensure you have the correct permissions to use the APIs](#). Your Veracode user account must have API permissions to be able to access and use the APIs.

Veracode REST APIs

The Veracode REST APIs follow the OpenAPI industry standard specification. These APIs return JSON instead of XML, and require authentication using [HMAC](#). See [Using the Veracode REST APIs](#) on page 42.

You can access Veracode REST APIs using a tool that supports Veracode HMAC authentication, including the [Java](#) or [Python](#) authentication libraries.



NOTE: Using the Veracode REST APIs requires the use of the [Veracode API ID and Key credentials](#).

Veracode XML APIs

The Veracode XML APIs are web APIs, each having a defined set of HTTP request messages that return structured response messages in XML. Some previous knowledge of how APIs function is recommended. See [Using the Veracode XML APIs](#) on page 68.

API Usage and Access Guidelines

Veracode supports two types of users that can use the Veracode APIs:

- Non-human user accounts, which cannot use the Veracode Platform
- Human user accounts, which use the Veracode Platform.

To avoid session conflicts, you must use the Veracode [API ID and key credentials](#) for human user accounts.

For information about these types of API users, read:

- [Understanding API Access](#)
- [Managing API Users](#)

Veracode requires that you generate Veracode API ID and key credentials to provide improved security when accessing APIs. You cannot use these credentials with cURL, but you can use them with the [HTTPIe command-line tool](#) as well as Veracode plugins and extensions.

See the following sections for more information about the Veracode API credentials:

- [Generating API ID and Key Credentials](#) on page 10
- [Using API ID and Key Credentials with the Veracode Plugins](#)

Veracode provides wrappers for Java and C# to simplify the use of the Veracode XML APIs. See [Using API Wrappers](#).

Understanding API Access

To be able to access the Veracode APIs, you must either have a human or non-human API customer account and the associated user roles for specific API tasks.

API user roles and permissions are determined in the user account administration page, and different APIs require different permissions. To use the Admin or Archer Report XML APIs you must have the API Account checkbox selected (non-human account) and the respective API user roles selected in the Login Settings section of the user account administration page.

Be sure that your IP address is in the list or range of addresses in the Allowed IP Addresses field of your user account login settings. If the IP range is set incorrectly, edit the [Allowed IP Addresses](#) field to include the IP address of the location of your login.



You can restrict API non-human users to teams, limiting their access to only data for applications that are associated with that team. Select Restrict to Selected Teams, and then choose the appropriate team. You can also restrict users to scan types, limiting them to performing static, dynamic, or Manual Penetration Testing scans.



If you intend to use the [Admin API](#) to create a new human user account, you have to pass the role parameters as well as the scan type permissions. The human user role parameters (case-sensitive) are Administrator, Creator, Executive, Mitigation Approver, Policy Administrator, Reviewer, Security Lead, Submitter, Security Insights, Veracode eLearning. The scan permission types are: Static Scan, Dynamic Scan, Manual Scan, Discovery Scan, DynamicMP Scan or All Scan Types.

It is important to note that when an application has its visibility set to [Teams & Security Leads](#), then a human account with the Reviewer, Creator, or Submitter user role must be a member of the [specified team](#) to be able to access that application using the APIs.

API Roles

The following tables outline which non-API user roles you must have to use APIs with a human user account to be able to automate specific tasks. If you are a member of a team, your permissions are also determined by the access of that team to specific accounts.

To use the Upload, Results, and Mitigation and Comments APIs, you must select one of the following checkboxes:

- API Account checkbox (non-human account) and the respective API user roles, or
- The respective non-API user roles (human account), such as Reviewer or Security Lead.

Archer Report API

API Role	Human Account User Role	Tasks
Archer Report	Submitter	<ul style="list-style-type: none"> • Run Archer reports • View reports

Admin API

If you intend to use the [Admin API](#) to create a new human user account, you have to pass the role parameters as well as the scan type permissions.



NOTE: The human user role parameters are case-sensitive.

The user role parameters are:

- Administrator
- Creator
- Executive
- Mitigation Approver
- Policy Administrator
- Reviewer
- Security Lead
- Submitter
- Security Insights
- eLearning

The scan permission types are:

- Static Scan
- Dynamic Scan
- Manual Scan
- Discovery Scan
- DynamicMP Scan
- All Scan Types



NOTE: When an application has its visibility set to [Teams & Security Leads](#), then a human account with the Reviewer, Creator, or Submitter user role must be a member of the [specified team](#) to be able to access that application using the APIs.

API Role	Human Account User Role	Tasks
Admin	Security Lead, Creator, or Submitter, depending on the task you want to perform.	<ul style="list-style-type: none"> • Create login account • Access Admin API

API Role	Human Account User Role	Tasks
		<ul style="list-style-type: none"> Delete team Create a curriculum Application portfolio Manage account level Elearning Assign application to any team Assign application to team Edit team Create team Edit login account Delete login account

Greenlight API

The Greenlight API User role is only available to organizations with active Veracode Greenlight subscriptions.

API Role	Human Account User Role	Tasks
Greenlight API User	Greenlight IDE User	<ul style="list-style-type: none"> Submit code for Greenlight scans Review Greenlight scan results

Mitigation and Comments API

API Role	Human Account User Role	Tasks
Mitigation and Comments	Mitigation Approver AND either Reviewer or Security Lead	Approve or reject proposed mitigations
Mitigation	Reviewer or Security Lead	<ul style="list-style-type: none"> View results Update results Approve or reject proposed mitigations

Results API

API Role	Human Account User Role	Tasks
Results	Reviewer or Security Lead	<ul style="list-style-type: none"> View reports View results Export custom data View the list of sandboxes Access Results API Download build and application results data,

API Role	Human Account User Role	Tasks
		and summary and detailed reports

Upload and Scan API

API Role	Human Account User Role	Tasks
Upload and Scan	<p>Security Lead, Creator, or Submitter, depending on the task you want to perform.</p> <p>A user with the Creator role can only create application profiles for teams in which the Creator is a member. The Submitter role can submit a scan request. The Security Lead role can perform all tasks. API users need the Upload and Scan API role to create a new application using Veracode Static for Visual Studio and to create sandboxes using the Veracode Jenkins Plugin.</p>	<ul style="list-style-type: none"> • Ability to enable applications for next day consultations for Creation and Update • Change application assurance level • Delete a sandbox scan • Create a sandbox scan for an application • Change the Archer name of an application • Manage policies • Create a sandbox in an application • View the list of sandboxes in an application • Create a policy scan for an application • Create a new application • Delete an application • Delete a policy scan • Use the Dynamic Analysis REST API

Upload API - Submit Only

This role can also create and delete scan requests, and has the ability to edit builds before rescanning the application. However, this role does not allow users to create new applications, including users of the Veracode integrations.

API Role	Human Account User Role	Tasks
Upload - Submit only	Submitter	<ul style="list-style-type: none"> • Create a new build for an existing application profile • Upload files to a build • Begin prescan • Check prescan status • Submit scan request • Delete a policy scan • Delete a sandbox scan • Create a policy scan

API Role	Human Account User Role	Tasks
		<ul style="list-style-type: none"> Create a sandbox scan View the list of sandboxes

Generating API ID and Key Credentials

Veracode provides API users with additional secure credentials.

Veracode API user accounts access APIs and plugins using [API ID and key credentials](#). The Veracode API ID and key authentication method provides improved security and session management for accessing the APIs.

You can generate and use the API ID and key credentials for both human and non-human API user accounts. If you use single sign-on with SAML, you can use the API ID and key credentials instead of a separate Veracode Platform API user account to access the APIs.

After you create API ID and key credentials, you can use these credentials to automatically sign in to Veracode APIs and plugins without using a separate API user account. You can only have one API ID and key pair at a time per Veracode user. If you create new credentials, Veracode automatically revokes the previous credentials. An administrator can revoke user credentials at any time.

Veracode sends an email notifying you when your API credentials are expiring one week before the expiration date and another one the day before the expiration date.

To use API credentials, Veracode recommends you use the [Veracode API wrappers](#), [HTTPIe with the appropriate Veracode Authentication Library](#), or one of the [Veracode IDE plugins](#).



NOTE: Veracode does not support using cURL from the command line to access Veracode APIs.

Generate Veracode API Credentials

Veracode API customers can access APIs and plugins using the ID and key credentials. If you use single sign-on with SAML, you can use the ID and key credentials instead of having to use a separate Veracode Platform API user account to access the APIs.

To create the API ID and key credentials:

- 1 Log in to the Veracode Platform.
- 2 Go to the user account menu and select **API Credentials**.
- 3 Click **Generate API Credentials**.
- 4 Copy the ID and secret key to a secure place to use them when logging into plugins.



You can only see these credentials this one time. You have the choice of setting them as environment variables or putting them in a credentials file. When you leave this page you cannot review your current credentials. The creation of new credentials revokes any old credentials after 24 hours. You can always [revoke the API credentials](#), if necessary. They expire in one calendar

year. If you want to extend the credentials beyond the expiration date, contact Veracode Technical Support at support@veracode.com.

After you create API ID and key credentials, you can use these credentials to automatically sign in to Veracode APIs and plugins without using a separate API user account to be able to access the APIs.

Revoke Veracode API Credentials

You can always revoke the API credentials, if necessary. They expire in one calendar year unless a Veracode administrator resets the expiration date.

The credentials expire after one calendar year by default unless they are revoked or set to another expiration date by your Veracode administrator. To revoke the API credentials:

- 1 Log in to the Veracode Platform.
- 2 Go to the user account menu and select **API Credentials**.
- 3 Click **Revoke API Credentials**.

The revoked credentials expire immediately.

Configuring the API Credentials File

Some Veracode products require that you provide your Veracode API credentials using a credentials file.

You can create the API credentials file on Windows, macOS, Linux, or UNIX. You add the file to a specific directory on the system running the Veracode integration. The integration, API wrapper, or command-line tool reads the file from this directory to access your credentials.

For macOS, Linux, and UNIX, you can alternately store your credentials in environment variables. You can use either the credentials file or environment variables, but not both.

Configure the API Credentials File on Windows

Before configuring the API credentials file, you must meet these prerequisites:

- You have generated, and have access to, your Veracode [API ID and key](#) credentials.
- Depending on your Veracode integration, you have added your API credentials to the [default] application profile. If you have an existing [greenlight] profile, delete it or rename it to [default].

To configure the API credentials file on Windows:

- 1 In File Explorer, open the `C:\Users` folder and, then, open your user directory.



- 2 Add a new folder to your user folder called `.veracode.` adding a final period to the name. The extra period at the end of the folder name specifies to File Explorer that you want to create a folder that starts with a period. The second period disappears after you create the folder.



- 3 Open a text editor, such as Notepad, and add your Veracode API credentials to a new file in the following format:

```
[default]
veracode_api_key_id = <your_api_key_id>
veracode_api_key_secret = <your_api_secret_key>
```

- 4 Choose **File > Save As** and enter "credentials" in the **File name** field. Including the quotation marks ensures that Notepad does not add a file extension. If you create the file with a file extension, right-click the file, choose **Properties**, and remove the file extension in the dialog.



- 5 Click **Save**.
- 6 Optionally, you can use the Advanced Attributes dialog in Windows to enable the **Encrypt contents to secure data** option on the credentials file. Enabling this option restricts user access to the file.

Configure the API Credentials File on macOS and Linux

Before configuring the API credentials file, you must meet these prerequisites:

- You have generated, and have access to, your Veracode [API ID and key](#) credentials.
- Depending on your Veracode integration, you have added your API credentials to the [default] application profile. If you have an existing [greenlight] profile, delete it or rename it to [default].

To configure the API credentials file on macOS, Linux, or UNIX:

- 1 On the command line, navigate to the folder for your user name. For example, \$HOME.
- 2 Run the following command to create the .veracode folder: `$ mkdir .veracode`
- 3 Navigate to the .veracode folder. For example: `$ cd .veracode`



NOTE: macOS immediately hides the .veracode folder because the folder name begins with a period.

- 4 Run the following touch command to create the credentials file: `$ touch credentials`
- 5 Open the Finder to .veracode.
- 6 Double-click the credentials file to open it in a text editor.
- 7 Copy the following text and paste it into the credentials file.

```
[default]
veracode_api_key_id = Your API ID
veracode_api_key_secret = Your API key
```

- 8 Replace the values with your API credentials.
- 9 Save the credentials file.
- 10 Optionally, you can run the following command to restrict access to the credentials file: `chmod 600 ~/.veracode/credentials`

Configure the API Credentials as Environment Variables on macOS and Linux

Storing your Veracode API credentials is a one-time task, automating authentication with Veracode APIs, API wrappers, extensions, and plugins.

Before configuring the API credentials file, you must meet these prerequisites:

- You have generated, and have access to, your Veracode [API ID and key](#) credentials.
- Depending on your Veracode integration, you have added your API credentials to the [default] application profile. If you have an existing [greenlight] profile, delete it or rename it to [default].

To store your Veracode API credentials as macOS, Linux, or UNIX environment variables:

From the command line, enter:

```
$ export VERACODE_API_KEY_ID=<your_api_key_id>
```

```
$ export VERACODE_API_KEY_SECRET=<your_api_secret_key>
```

Enabling HMAC Authentication

The Veracode integrations and APIs use HMAC authentication when accessing API resources.

About HMAC Authentication

Veracode API authentication uses an API ID and key that you can generate for any Veracode user account, regardless of type (human or API user) and login method (username/password or SAML). The API ID and key digitally sign the HTTP header accompanying an API request with Hash-based Message Authentication Code (HMAC). This process provides added security:

- Credentials are not sent in the clear as plain text. The API key is never transmitted, but encrypts the HMAC from the sender-side and decrypts it from the server-side.
- The HMAC signature validates that the message was not tampered with or altered in transit. Any change to the message invalidates the HMAC.
- The HMAC signature includes a nonce (one-time code) that prevents replay attacks.
- You can revoke and regenerate the API ID and key to respond to an accidental credentials leak.

Prerequisites

Ensure you have followed the steps in:

- [Generating API ID and Key Credentials](#) on page 10
- [Configuring the API Credentials File](#) on page 11

HMAC Authentication for the API Wrappers

The [API wrappers](#) provide the easiest way for occasional users to achieve secure API functionality.

After you generate and store your Veracode API credentials, the Java and C# API wrappers are enabled for HMAC authentication and ready for use from the command line and in your code.

As a first step in using HMAC authentication, incorporate the Java or C# wrapper into your automation. You can use the wrappers as a command-line tool and supply the API ID and key, or store them in a credentials file. The other HMAC steps are then automatic.

See [Using the API Wrappers as a Library or a Command-Line Application](#) for information about using the API wrappers in plugins.

The API wrappers are also the best way to troubleshoot your Veracode environment.

The HMAC signing example programs for [Java](#) and [C#](#) use the Java and C# API wrappers.



NOTE: If you use the API wrappers, ensure that you always run the latest version.

HMAC Authentication for Using the XML APIs from the Command Line

The cURL command-line tool does not support HMAC authentication, therefore Veracode provides support for the HTTPie command-line tool. To use HTTPie and HMAC authentication with the XML APIs:

- Download and install the Python programming language. Veracode recommends Python 3.5 or later. If you have a recommended version, you can omit this step. Otherwise, refer to the [Python Wiki](#) for advice on choosing a Python download.
- [Install the Python Authentication Library](#).
- Download and install the HTTPie command-line tool by following the steps in [Using HTTPie with the Python Authentication Library](#) on page 16

HMAC Authentication for Python Programming

To use HMAC authentication with Python programs:

- 1 [Install the Python Authentication Library](#).
- 2 Review the [HMAC Signing Example in Python](#) on page 17.

HMAC Authentication for Java Programming

To use HMAC authentication with Java programs:

- 1 [Install the Java Authentication Library](#).
- 2 Review the [HMAC Signing Example in Java](#) on page 17.

HMAC Authentication for C# Programming

To use HMAC authentication with C# programs, review the [HMAC Signing Example in C#](#) on page 21.

Troubleshooting Tips

Issues that prevent HMAC from working correctly, or make it appear that HMAC is not working at all, include:

- Incorrect credentials. The most frequent problem encountered after setting up HMAC authentication is incorrect API ID and key. For example, you may have multiple accounts and associate the wrong set of credentials with the account you are setting up. Ensure credential

sets are current and secure. If your system is not working, try [revoking the existing credentials](#), creating new credentials, and retrying.

- Incorrect account type or [user roles](#). Each API specifies what account type (human user or non-human API user) and user roles are required to call it. A role or account error can sometimes be misunderstood as a larger problem with authentication. Check the specific API call reference page in the Veracode Help Center for required account and role information.
- Problems connecting to the Veracode Platform. As a test, run the [getmaintenancescheduleinfo.do](#) XML API from the command line. Non-human user accounts require the [Admin API](#) role to use this call. Human user accounts require the [Administrator](#) role to use this call.

```
http --auth-type=veracode_hmac "https://  
analysiscenter.veracode.com/api/3.0/getmaintenancescheduleinfo.do"
```

You should quickly get a short response.

- Inaccurate system time. Although infrequent, HMAC authentication fails if the system time of the client and server are substantially out-of-sync. Compare your system time with actual time at [time.is](#) to ensure your system time is close to actual time.

For more help with HMAC authentication issues, contact Veracode Technical Support at support@veracode.com.

Install the Java Authentication Library

If you want to use the Veracode APIs with a Java application, you must download and install the Java authentication library.

To install the Java authentication library:

- 1 Download the [Veracode API-signing Java library](#).
- 2 Store the downloaded JAR file in your application project directory.

Install the Python Authentication Library

You install the Python authentication library to enable HMAC authentication. HMAC is a required security measure for using the Veracode APIs with your Python applications.

Before installing the Python authentication library, you must meet these prerequisites:

- A Linux, UNIX, macOS, or Windows machine.
- Installed Python 3.5 or later. If you need to install or upgrade Python, refer to the [Python Wiki](#) for advice on choosing a Python download.
- Upgraded to the latest version of pip, the Python package manager. If you have a new installation of a compatible Python version, you can omit this step. Otherwise, you must upgrade pip to work with Veracode authentication libraries. You can read pip upgrade instructions on the [pip website](#). Look for the upgrade commands in the installation documentation.
 - If using Ubuntu, you have uninstalled any system-installed version of pip, installed the latest version of pip, and uninstalled any system-installed version of HTTPie, as described in [Preparing Ubuntu for the Python Authentication Library Installation](#) on page 16.
- Generated [Veracode API credentials](#).
- [Stored your Veracode API credentials](#).

To download and install the Python authentication library from the Python Package Index (PyPI), run the following command:

```
pip install veracode-api-signing
```

Preparing Ubuntu for the Python Authentication Library Installation

Before installing the Python authentication library on Ubuntu, Veracode recommends that you uninstall any system-installed version of pip and, then, install the latest version of pip. Veracode also recommends that you uninstall any system-installed version of HTTPie.

Run the following command to uninstall a system-installed version of pip:

```
sudo apt-get remove python-pip
```

Run the following command to install the latest version of pip:

```
sudo easy_install pip
```

Run the following command to uninstall a system-installed version of HTTPie:

```
sudo apt-get purge httpie
```

Next step

You can now install the Python authentication library. See [Install the Python Authentication Library](#) on page 15.

Using HTTPie with the Python Authentication Library

The Veracode APIs require that you enable HMAC authentication for your Python application as a security measure for accessing API resources. The Python authentication library provides an integration between HTTPie and the Veracode APIs, which adds HMAC authentication when using the Veracode APIs from the command line.

You use the library to:

- Load the API credentials
- Generate an HMAC authorization header
- Issue an HTTP call to a Veracode API with a valid endpoint

The default HTTP method is `GET`. For command examples, see [Using the Veracode REST APIs](#) or [Using the Veracode XML APIs](#).

To download and install HTTPie, see <https://httpie.org>.

To learn more about HTTPie, you can run `http --help` and review the HTTPie documentation. You can also get tips and examples from <https://devhints.io/httpie>.

HMAC authentication is the same for all calls, but the other aspects of authentication are specific to the API endpoint you want to call.

HMAC Signing Example in Python

Veracode provides a Python library you can use to enable HMAC signing within your application.

For an example of how you can use this library, go to: <https://github.com/veracode/veracode-python-hmac-example>.

HMAC Signing Example in Java

This is a Java example of how to enable HMAC signing within your application. The example implementation of the HMAC signing algorithm allows you to authenticate with the Veracode APIs.

HmacRequestSigner.java

```
package com.veracode.hmac_request_signing;

import java.io.UnsupportedEncodingException;
import java.net.URL;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.util.Locale;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

public final class HmacRequestSigner {

    // Included in the signature to inform Veracode of the signature
    // version.
    private static final String VERACODE_REQUEST_VERSION_STRING =
        "vcode_request_version_1";

    // Expected format for the unencrypted data string.
    private static final String DATA_FORMAT = "id=%s&host=%s&url=
    %s&method=%s";

    // Expected format for the Authorization header.
    private static final String HEADER_FORMAT = "%s id=%s,ts=%s,nonce=
    %s,sig=%s";

    // Expect prefix to the Authorization header.
    private static final String VERACODE_HMAC_SHA_256 = "VERACODE-
    HMAC-SHA-256";

    // HMAC encryption algorithm.
    private static final String HMAC_SHA_256 = "HmacSHA256";

    // Charset to use when encrypting a string.
    private static final String UTF_8 = "UTF-8";

    // A cryptographically secure random number generator.
    private static final SecureRandom secureRandom = new
    SecureRandom();

    // Private constructor.
    private HmacRequestSigner() {
        /*
         * This is a utility class that should only be accessed
```

```

through its
    * static methods.
    */
}

/**
 * Entry point for HmacRequestSigner. Returns the value for the
 * Authorization header for use with Veracode APIs when provided
an API id,
 * secret key, and target URL.
 *
 * @param id
 *         An API id for authentication
 * @param key
 *         The secret key corresponding to the API id
 * @param url
 *         The URL of the called API, including query
parameters
 *
 * @return The value to be put in the Authorization header
 *
 * @throws UnsupportedOperationException
 * @throws IllegalStateException
 * @throws NoSuchAlgorithmException
 * @throws InvalidKeyException
 */
public static String getVeracodeAuthorizationHeader(final String
id, final String key, final URL url, final String httpMethod)
    throws InvalidKeyException, NoSuchAlgorithmException,
IllegalStateException, UnsupportedOperationException {
    final String urlPath = (url.getQuery() == null) ?
url.getPath() : url.getPath().concat("?").concat(url.getQuery());
    final String data = String.format(DATA_FORMAT, id,
url.getHost(), urlPath, httpMethod);
    final String timestamp =
String.valueOf(System.currentTimeMillis());
    final String nonce =
DatatypeConverter.printHexBinary(generateRandomBytes(16)).toLowerCase(
Locale.US);
    final String signature = getSignature(key, data, timestamp,
nonce);
    return String.format(HEADER_FORMAT, VERACODE_HMAC_SHA_256,
id, timestamp, nonce, signature);
}

/*
 * Generate the signature expected by the Veracode platform by
chaining
 * encryption routines in the correct order.
 */
private static String getSignature(final String key, final String
data, final String timestamp, final String nonce)
    throws InvalidKeyException, NoSuchAlgorithmException,
IllegalStateException, UnsupportedOperationException {
    final byte[] keyBytes = DatatypeConverter.parseHexBinary(key);
    final byte[] nonceBytes =
DatatypeConverter.parseHexBinary(nonce);
    final byte[] encryptedNonce = hmacSha256(nonceBytes,
keyBytes);
    final byte[] encryptedTimestamp = hmacSha256(timestamp,
encryptedNonce);

```

```

        final byte[] signingKey =
            hmacSha256(VERACODE_REQUEST_VERSION_STRING, encryptedTimestamp);
        final byte[] signature = hmacSha256(data, signingKey);
        return
            DatatypeConverter.printHexBinary(signature).toLowerCase(Locale.US);
    }

    // Encrypt a string using the provided key.
    private static byte[] hmacSha256(final String data, final byte[]
key)
        throws NoSuchAlgorithmException, InvalidKeyException,
IllegalStateException, UnsupportedEncodingException {
        final Mac mac = Mac.getInstance(HMAC_SHA_256);
        mac.init(new SecretKeySpec(key, HMAC_SHA_256));
        return mac.doFinal(data.getBytes(UTF_8));
    }

    // Encrypt a byte array using the provided key.
    private static byte[] hmacSha256(final byte[] data, final byte[]
key)
        throws NoSuchAlgorithmException, InvalidKeyException {
        final Mac mac = Mac.getInstance(HMAC_SHA_256);
        mac.init(new SecretKeySpec(key, HMAC_SHA_256));
        return mac.doFinal(data);
    }

    // Generate a random byte array for cryptographic use.
    private static byte[] generateRandomBytes(final int size) {
        final byte[] key = new byte[size];
        secureRandom.nextBytes(key);
        return key;
    }
}

```

Main.java

```

package com.veracode.hmac_request_signing;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import java.security.InvalidKeyException;
import java.security.KeyManagementException;
import java.security.NoSuchAlgorithmException;
import java.security.cert.X509Certificate;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

import org.json.JSONException;
import org.json.JSONObject;

public class Main {

```

```

        private static final String URL_BASE = "api.veracode.com";
        private static final String URL_PATH = "/appsec/v1/applications/";
        private static final String GET = "GET";
        private static final String APP_GUID =
"8b86411e-65f9-4224-948a-64559c777d10";
        private static final String ACCESS_KEY_ID =
"d5b6f2a2ed0b6890bbd32e949f72c8c8";
        private static final String SECRET_ACCESS_KEY =
"530da152f87e5530c82f786907fbc74b09a6894785a78bab3891632ba69325400a407
13bdc11d2a6d2d1c3969431281c0a73f455a53c0ed5ea0756e9c54f366c";

        /**
         * The main method for our demo. This makes a simple API call
         using our example HMAC signing class
         * and writes the response to the output stream.
         *
         * @param args command line arguments - ignored
         */
        public static void main(final String[] args) {
            try {
                /**
                 * Combine the URL base with the specific URL endpoint we wish to
                 access.
                 * This is REST, so the GUID we are accessing is in the URL.
                 */
                final URL applicationsApiUrl = new URL("https://" + URL_BASE +
URL_PATH + APP_GUID);

                /**
                 * Now we use the url above and our example HMAC signer class to
                 generate a Veracode HMAC header for later use.
                 */
                final String authorizationHeader =
HmacRequestSigner.getVeracodeAuthorizationHeader(ACCESS_KEY_ID,
SECRET_ACCESS_KEY, applicationsApiUrl, GET);

                /**
                 * Here we are using Java built in HTTPS protocols to handle
                 making a call to the API's URL.
                 * We also set the request method to GET.
                 */
                final HttpsURLConnection connection = (HttpsURLConnection)
applicationsApiUrl.openConnection();
                connection.setRequestMethod(GET);

                /**
                 * This is where we add the Authorization header with the value
                 returned by our example HMAC signer class.
                 */
                connection.setRequestProperty("Authorization",
authorizationHeader);

                /**
                 * Now we just need to make the actual call by opening up the
                 response stream and read from it.
                 */
                try (InputStream responseInputStream =
connection.getInputStream()) {
                    readResponse(responseInputStream);
                }
            }
        }

```

```

    } catch (InvalidKeyException | NoSuchAlgorithmException |
IllegalStateException | IOException e) {
        e.printStackTrace();
    }
}

/*
 * A simple method to read an input stream (containing JSON) to
System.out.
 */
private static void readResponse(InputStream responseInputStream)
throws IOException, JSONException {
    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    byte[] responseBytes = new byte[16384];
    int x = 0;
    while ((x = responseInputStream.read(responseBytes, 0,
responseBytes.length)) != -1) {
        outputStream.write(responseBytes, 0, x);
    }
    outputStream.flush();
    System.out.println((new
JSONObject(outputStream.toString())).toString(4));
}
}

```

HMAC Signing Example in C#

This is a C# example of how to enable HMAC signing within your application shows how to authenticate when using the Veracode APIs.

You can download the code for this example from tools.veracode.com/integrations/Microsoft/Microsoft/VisualStudio/update/Veracode.HmacExample.zip.

HmacAuthHeader.cs

```

using System;
using System.Runtime.Remoting.Metadata.W3cXsd2001;
using System.Security.Cryptography;
using System.Text;
using System.Text.RegularExpressions;

namespace Veracode.HmacExample.App
{
    public abstract class HmacAuthHeader
    {
        private static readonly RNGCryptoServiceProvider RngRandom =
new RNGCryptoServiceProvider();

        public static readonly HmacAuthHeader HmacSha256 = new
HmacSha256AuthHeader();

        private sealed class HmacSha256AuthHeader : HmacAuthHeader
        {
            protected override string GetHashAlgorithm() { return
"HmacSHA256"; }

            protected override string GetAuthorizationScheme() { return

```

```

"VERACODE-HMAC-SHA-256"; }

    protected override string GetRequestVersion() { return
"vcode_request_version_1"; }

    protected override string GetTextEncoding() { return "UTF-8"; }

    protected override int GetNonceSize() { return 16; }

    internal HmacSha256AuthHeader() { }

    protected abstract string GetHashAlgorithm();
    protected abstract string GetAuthorizationScheme();
    protected abstract string GetRequestVersion();
    protected abstract string GetTextEncoding();
    protected abstract int GetNonceSize();

    protected string CurrentDateStamp()
    {
        return ((long)((TimeSpan)(DateTime.UtcNow - new DateTime(1970,
1, 1))).TotalMilliseconds).ToString();
    }

    protected byte[] NewNonce(int size)
    {
        byte[] nonceBytes = new byte[size];
        RngRandom.GetBytes(nonceBytes);

        return nonceBytes;
    }

    protected byte[] ComputeHash(byte[] data, byte[] key)
    {
        HMAC mac = HMAC.Create(GetHashAlgorithm());
        mac.Key = key;

        return mac.ComputeHash(data);
    }

    protected byte[] CalculateDataSignature(byte[] apiKeyBytes,
byte[] nonceBytes, string dateStamp, string data)
    {
        byte[] kNonce = ComputeHash(nonceBytes, apiKeyBytes);
        byte[] kDate =
ComputeHash(Encoding.GetEncoding(GetTextEncoding()).GetBytes(dateStamp
), kNonce);
        byte[] kSignature =
ComputeHash(Encoding.GetEncoding(GetTextEncoding()).GetBytes(GetReques
tVersion()), kDate);

        return
ComputeHash(Encoding.GetEncoding(GetTextEncoding()).GetBytes(data),
kSignature);
    }

    public string CalculateAuthorizationHeader(string apiId,
string apiKey, string hostName, string uriString, string
urlQueryParams, string httpMethod)
    {
        try

```

```

        {
            if (urlQueryParams != null)
            {
                uriString += (urlQueryParams);
            }
            string data =
$"id={apiId}&host={hostName}&url={uriString}&method={httpMethod}";
            string dateStamp = CurrentDateStamp();
            byte[] nonceBytes = NewNonce(GetNonceSize());
            byte[] dataSignature =
CalculateDataSignature(FromHexBinary(apiKey), nonceBytes, dateStamp,
data);
            string authorizationParam =
$"id={apiId},ts={dateStamp},nonce={ToHexBinary(nonceBytes)},sig={ToHex
Binary(dataSignature)}";

            return GetAuthorizationScheme() + " " + authorizationParam;
        }
        catch (Exception e)
        {
            throw new Exception(e.Message, e);
        }
    }

    public static string ToHexBinary(byte[] bytes)
    {
        return new SoapHexBinary(bytes).ToString();
    }

    public static byte[] FromHexBinary(string hexBinaryString)
    {
        return SoapHexBinary.Parse(hexBinaryString).Value;
    }

    public static bool IsValidHexBinary(string hexBinaryString)
    {
        {
            if (hexBinaryString != null)
            {
                try
                {
                    byte[] bytes = FromHexBinary(hexBinaryString);
                    return bytes != null;
                }
                catch (Exception) { }
            }
        }

        return false;
    }

    public static bool IsValidAuthHeaderToken(string
authHeaderToken)
    {
        {
            if (authHeaderToken != null)
            {
                // For valid Authorization header token syntax see https://
www.ietf.org/rfc/rfc2617.txt, https://www.ietf.org/rfc/rfc2068.txt
                bool isMatch = Regex.IsMatch(authHeaderToken, "^[\x21\x23-\
\x27\x2A-\x2B\x2D-\x2E\x30-\x39\x41-\x5A\x5E-\x7A\x7C\x7E]
+$");
            }
        }

        return isMatch;
    }

```

```

    }

    return false;
}

private HmacAuthHeader() { }
}
}

```

Program.cs

```

using System;
using System.Net;

namespace Veracode.HmacExample.App
{
    public class Program
    {
        private const string AuthorizationHeader = "Authorization";
        private const string ApiId = "VERACODE_API_ID_GOES_HERE";
        private const string ApiKey = "VERACODE_SECRET_KEY_GOES_HERE";

        public static void Main(string[] args)
        {
            try
            {
                const string urlBase = "analysiscenter.veracode.com";
                const string urlPath = "/api/5.0/getapplist.do";
                var urlParams = string.Empty;
                const string httpVerb = "GET";

                var webClient = new WebClient
                {
                    BaseAddress = $"https://{urlBase}"
                };

                var authorization =
                    HmacAuthHeader.HmacSha256.CalculateAuthorizationHeader(ApiId, ApiKey,
                        urlBase, urlPath, urlParams, httpVerb);

                webClient.Headers.Add(AuthorizationHeader, authorization);

                var result = webClient.DownloadString(urlPath);

                Console.WriteLine(result);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
            finally
            {
                Console.WriteLine("Press any key to continue.");
                Console.ReadKey();
            }
        }
    }
}

```


}

Veracode XML API Quick Reference Guide

This table lists the Veracode XML API calls and their parameters. More detailed information is available in the Help Center.

Table 1:

XML API Call	Location	Required Parameters	Optional Parameters	Scan Type
Upload API Calls				
beginprescan.do	https://analysiscenter.veracode.com/api/5.0/beginprescan.do	app_id	sandbox_id, auto_scan, scan_all_nonfatal_top_level_modules	Static
beginscan.do	https://analysiscenter.veracode.com/api/5.0/beginscan.do	app_id and one of the following: <ul style="list-style-type: none"> modules scan_all_top_level_modules scan_previously_selected_modules scan_selected_modules 	sandbox_id	Static
createapp.do	https://analysiscenter.veracode.com/api/5.0/createapp.do	app_name, business_criticality	description, vendor_id, policy, business_unit, business_owner, business_owner_email, teams, origin, industry, app_type, deployment_method, web_application, archer_app_name, tags, next_day_scheduling_enabled	Static, Dynamic
createbuild.do	https://analysiscenter.veracode.com/api/5.0/createbuild.do	app_id, version	platform, lifecycle_stage, launch_date, sandbox_id,	Static, Dynamic

XML API Call	Location	Required Parameters	Optional Parameters	Scan Type
			legacy_scan_engine	
deleteapp.do	https://analysiscenter.veracode.com/api/5.0/deleteapp.do	app_id		Static, Dynamic
deletebuild.do	https://analysiscenter.veracode.com/api/5.0/deletebuild.do	app_id	sandbox_id	Static, Dynamic
getappinfo.do	https://analysiscenter.veracode.com/api/5.0/getappinfo.do	app_id		Static, Dynamic
getapplist.do	https://analysiscenter.veracode.com/api/5.0/getapplist.do		include_user_info	Static, Dynamic
getbuildinfo.do	https://analysiscenter.veracode.com/api/5.0/getbuildinfo.do	app_id	build_id, sandbox_id	Static, Dynamic
getbuildlist.do	https://analysiscenter.veracode.com/api/5.0/getbuildlist.do	app_id	sandbox_id	Static, Dynamic
getfilelist.do	https://analysiscenter.veracode.com/api/5.0/getfilelist.do	app_id	build_id, sandbox_id	Static
getpolicylist.do	https://analysiscenter.veracode.com/api/5.0/getpolicylist.do			Static, Dynamic
getprescanresults.do	https://analysiscenter.veracode.com/api/5.0/getprescanresults.do	app_id	build_id, sandbox_id	Static
getvendorlist.do	https://analysiscenter.veracode.com/api/5.0/getvendorlist.do			Static, Dynamic

XML API Call	Location	Required Parameters	Optional Parameters	Scan Type
removefile.do	https://analysiscenter.veracode.com/api/5.0/removefile.do	app_id, file_id	sandbox_id	Static
updateapp.do	https://analysiscenter.veracode.com/api/5.0/updateapp.do	app_id	app_name, description, policy, business_criticality, business_unit, business_owner, business_owner_email, teams, origin, industry, app_type, deployment_method, archer_app_name, tags, custom_field_name, custom_field_value, next_day_scheduling_enabled	Static, Dynamic
updatebuild.do	https://analysiscenter.veracode.com/api/5.0/updatebuild.do	app_id	build_id, version, lifecycle_stage, launch_date, sandbox_id	Static, Dynamic
uploadfile.do	https://analysiscenter.veracode.com/api/5.0/uploadfile.do	app_id, file	sandbox_id, save_as	Static
uploadlargefile.do	https://analysiscenter.veracode.com/api/5.0/uploadlargefile.do	app_id, file	filename, sandbox_id	Static
Results API Calls				
detailedreport.do	https://analysiscenter.veracode.com/api/5.0/detailedreport.do	build_id		Static, Dynamic, Manual
detailedreportpdf.do	https://analysiscenter.veracode.com/api/4.	build_id		Static, Dynamic, Manual

XML API Call	Location	Required Parameters	Optional Parameters	Scan Type
	0/ detailedreportpdf. do			
getaccountcusto mfieldlist.do	https:// analysiscenter.ve racode.com/api/5. 0/ getaccountcusto mfieldlist.do			Static, Dynamic, Manual
getappbuilds.do	https:// analysiscenter.ve racode.com/api/4. 0/getappbuilds.do		include_in_progre ss, only_latest, report_changed_ since	Static, Dynamic, Manual
getcallstacks.do	https:// analysiscenter.ve racode.com/api/5. 0/getcallstacks.do	build_id, flaw_id		Static
summaryreport.d o	https:// analysiscenter.ve racode.com/api/4. 0/ summaryreport.d o	build_id		Static, Dynamic, Manual
summaryreportpd f.do	https:// analysiscenter.ve racode.com/api/4. 0/ summaryreportpd f.do	build_id		Static, Dynamic, Manual
thirdpartyreportp df.do	https:// analysiscenter.ve racode.com/api/4. 0/ thirdpartyreportp df.do	build_id		Static, Dynamic, Manual
Admin API Calls				
createteam.do	https:// analysiscenter.ve racode.com/api/3. 0/createteam.do	team_name	members	—
createuser.do	https:// analysiscenter.ve racode.com/api/3. 0/createuser.do	first_name, last_name, email_address, roles	custom_id, is_saml_user, login_enabled, phone,	—

XML API Call	Location	Required Parameters	Optional Parameters	Scan Type
			requires_token, teams, title	
deleteteam.do	https://analysiscenter.veracode.com/api/3.0/deleteteam.do	team_id		—
deleteuser.do	https://analysiscenter.veracode.com/api/3.0/deleteuser.do	username	custom_id	—
getcurriculumlist.do	https://analysiscenter.veracode.com/api/3.0/getcurriculumlist.do			—
getmaintenancescheduleinfo.do	https://analysiscenter.veracode.com/api/3.0/getmaintenancescheduleinfo.do			—
getteamlist.do	https://analysiscenter.veracode.com/api/3.0/getteamlist.do			—
getteaminfo.do	https://analysiscenter.veracode.com/api/3.0/getteaminfo.do	team_id	include_users, include_applications	—
gettracklist.do	https://analysiscenter.veracode.com/api/3.0/gettracklist.do			—
getuserinfo.do	https://analysiscenter.veracode.com/api/3.0/getuserinfo.do	username	custom_id	—
getuserlist.do	https://analysiscenter.veracode.com/api/3.0/getuserlist.do		first_name, last_name, custom_id, email_address, login_account_type, phone, teams, roles,	—

XML API Call	Location	Required Parameters	Optional Parameters	Scan Type
			is_saml_user, login_enabled, requires_token, is_elearning_manager, elearning_track, elearning_curriculum, keep_elearning_active, custom_one, custom_two, custom_three, custom_four, custom_five	
updateteam.do	https://analysiscenter.veracode.com/api/3.0/updateteam.do	team_id	members, team_name	—
updateuser.do	https://analysiscenter.veracode.com/api/3.0/updateuser.do	username, custom_id	first_name, last_name, email_address, login_account_type, phone, teams, roles, is_saml_user, login_enabled, requires_token, has_ip_restrictions, allowed_ip_addresses is_elearning_manager, elearning_track, elearning_curriculum, keep_elearning_active, custom_one, custom_two, custom_three, custom_four, custom_five	—
Mitigation API Calls				
getmitigationinfo.do	https://analysiscenter.veracode.com/api/	build_id, flaw_id_list		Static

XML API Call	Location	Required Parameters	Optional Parameters	Scan Type
	getmitigationinfo.do			
updatemitigationinfo.do	https://analysiscenter.veracode.com/api/updatemitigationinfo.do	build_id, action, comment, flaw_id_list		Static
Archer API Calls				
archer.do	https://analysiscenter.veracode.com/api/archer.do		app_id, period, from_date, to_date, scan_type	Static, Dynamic, Manual
generatearcherreport.do	https://analysiscenter.veracode.com/api/2.0/generatearcherreport.do		app_id, period, from_date, to_date, scan_type	Static, Dynamic, Manual
downloadarcherreport.do	https://analysiscenter.veracode.com/api/2.0/downloadarcherreport.do		token	Static, Dynamic, Manual
DynamicDS API Calls				
createdynamicscan.do	https://analysiscenter.veracode.com/api/5.0/createdynamicscan.do	app_id	scan_name	Dynamic
dynamicscanconfig.do	https://analysiscenter.veracode.com/api/5.0/dynamicscanconfig.do	app_id, target_url, last_name, first_name, phone, email	https_http_inclusion, directory_restrictions_policy	Dynamic
dynamicincludeexclude.do	https://analysiscenter.veracode.com/api/5.0/dynamicincludeexclude.do	app_id, url	is_exclude, https_http_inclusion, directory_restrictions_policy	Dynamic

XML API Call	Location	Required Parameters	Optional Parameters	Scan Type
submitdynamicscan.do	https://analysiscenter.veracode.com/api/5.0/submitdynamicscan.do	app_id	start_time, end_time	Dynamic
rescandynamicscan.do	https://analysiscenter.veracode.com/api/5.0/rescandynamicscan.do	app_id	flaw_only, scan_name	Dynamic
getdynamicflaws.do	https://analysiscenter.veracode.com/api/5.0/getdynamicflaws.do	build_id, flaw_id		Dynamic
getdynamicstatus.do	https://analysiscenter.veracode.com/api/5.0/getdynamicstatus.do	app_id	build_id	Dynamic
uploadformbasedloginscript.do	https://analysiscenter.veracode.com/api/5.0/uploadformbasedloginscript.do	app_id, login_script, verification_url, verification_script		Dynamic
addbrowserbasedlogin.do	https://analysiscenter.veracode.com/api/5.0/addbrowserbasedlogin.do	app_id, username, password	windows_domain	Dynamic
assignvsa.do	https://analysiscenter.veracode.com/api/5.0/assignvsa.do	app_id, vsg_id		Dynamic
getvsalist.do	https://analysiscenter.veracode.com/api/5.0/getvsalist.do			Dynamic

XML API Call	Location	Required Parameters	Optional Parameters	Scan Type
	racode.com/api/5.0/getvsalist.do			
Flaw Report API Calls				
generateflawreport.do	https://analysiscenter.veracode.com/api/2.0/generateflawreport.do		token, scan_type	—
downloadflawreport.do	https://analysiscenter.veracode.com/api/2.0/downloadflawreport.do		app_id_list, scan_type	—
Sandbox API Calls				
createsandbox.do	https://analysiscenter.veracode.com/api/5.0/createsandbox.do	app_id, sandbox_name		—
getsandboxlist.do	https://analysiscenter.veracode.com/api/5.0/getsandboxlist.do	app_id		—
promotesandbox.do	https://analysiscenter.veracode.com/api/4.0/promotesandbox.do	build_id		—
updatesandbox.do	https://analysiscenter.veracode.com/api/4.0/updatesandbox.do	sandbox_id, custom_field_name, custom_field_value		—
deletesandbox.do	https://analysiscenter.veracode.com/api/5.0/deletesandbox.do	sandbox_id		—

XML API Call	Location	Required Parameters	Optional Parameters	Scan Type
VAST API Calls				
sharedreport.do	https://analysiscenter.veracode.com/api/3.0/sharedreport.do	app_id, shared_report_id		—
sharedreportpdf.do	https://analysiscenter.veracode.com/api/3.0/sharedreportpdf.do	app_id, shared_report_id		—
getsharedreportlist.do	https://analysiscenter.veracode.com/api/3.0/getsharedreportlist.do	app_id		—
getsharedreportinfo.do	https://analysiscenter.veracode.com/api/3.0/getsharedreportinfo.do	app_id	shared_report_id	—

Frequently Asked Questions (FAQ)

This section contains the answers to common questions about the Veracode XML APIs, plugins, and REST APIs.

XML API and Veracode Plugin FAQ

Question	Answer
Why does Veracode not use basic authentication with username and passwords for API integrations?	Veracode uses the Veracode API ID and key with HMAC signing because this method provides maximum protection against man-in-the-middle and session replay attacks.
Does Veracode Static for Eclipse work with other Eclipse-derived IDEs?	Veracode Static for Eclipse may run on Eclipse-derived IDEs like Spring Suite, but Veracode does not provide support for these IDEs.
How do I check prescan results in the API?	Prescans usually complete very quickly and you receive email notifications when they complete. If you want to check for prescan results using the Upload API, use the getprescanresults.do call.

Question	Answer
How do I use the API to query tags in the Results API?	If you want to query tags in applications, you can add unique tags as metadata when creating your applications. You can then query your applications based on any of the metadata. Use createapp.do to create an application with metadata. Use the following calls of the Results API to get the scan results of applications: getapplist.do to get the full list of your applications and then getappinfo.do to get information for a specific application, including any metadata, if applicable. To get a detailed report for any application, call detailedreport.do , which returns the results in an XML document.
How does Veracode ensure secure communication when making API calls to the Veracode Platform?	Using the Veracode API ID and key credentials ensures the most secure communication when using APIs. Security features include HMAC signatures to ensure the identity of the requester, a nonce to prevent replay attacks, and the ability to revoke API ID and key pairs if they are ever compromised. When using user credentials, Veracode uses TLS 1.2 or later for both the IDE plugins and for the Veracode XML APIs, which ensures that data transmitted between your client and the Veracode Platform is encrypted and secure.
How do I run an API scan if there are "unsupported frameworks" warnings in my prescan results?	If you want to ensure the scan completes even though there are non-fatal errors such as unsupported frameworks, ensure you use the <code>scan_all_top_level_modules</code> parameter when you use the beginscan.do call. Alternately, you can use <code>scan_selected_modules</code> , <code>scan_previously_selected_modules</code> , or modules with a list of module IDs, returned by the prescan.

REST API FAQ

Question	Answer
I want to use HMAC signing but I am not using Java. What are my options?	You can do one of the following: <ul style="list-style-type: none"> • Perform the external signing step on the command line using either the Java or Python tool or the Veracode Java or C# API wrappers. • Use one of the community-provided HMAC implementations at https://github.com/veracode/veracode-community-projects. • Enable HMAC signing within your C# application.

Troubleshooting

This section helps you remedy common problems and understand how better to use the Veracode XML APIs and plugins.

API	Issue	Solution
Any API	I received a HTTP 401 or Access Denied error. I do not have access to the APIs or I am unsure what kind of access I need.	You need to have specific API roles assigned to your user account by your Veracode administrator. See Understanding API Access for more information.
Any API	I cannot log in to Veracode when using the APIs.	Verify that your IP address is in the list or range of addresses in the Allowed IP Addresses field of your user account login settings. If the IP range is set incorrectly, edit the Allowed IP Addresses field to include the IP address of the location of your login.
Any API	The scan stopped after prescan.	To determine why a scan that started from an API failed after prescan, review the response code returned from the beginscan.do API call . When your script calls <code>beginscan.do</code> , the API returns a status code that confirms the scan successfully started, or provides an error message to explain why the scan did not start.
Any API	I received cURL error 35.	If you receive the cURL error 35: "Unknown SSL protocol error in connection to ...", you need to update your version of cURL. Alternatively, you can pass the option <code>-3</code> , which forces cURL to use SSL version 3 when negotiating with a remote SSL server.
Any Integration	<p>I have received one of the following messages:</p> <ul style="list-style-type: none"> Received fatal alert: handshake_failure Peer not authenticated error System.Net.WebException was unhandled. Message=The request was aborted: Could not create SSL/TLS secure channel OpenSSL::SSL::SSL_ERROR: Received fatal alert: handshake_failure The underlying connection was closed: An unexpected error occurred on a send. Could not create SSL/TLS secure channel 	If you are using an integration that attempts to connect over TLS 1.0, you may receive one of these error messages. Read more about our TLS support .
Archer API	Invalid IP address range.	Ensure that you are attempting to connect from an IP address that is allowed by the IP address restrictions for the login you are using.

API	Issue	Solution
Archer API	Invalid login type.	Ensure that you are providing credentials for an API class login with the Archer API role.
Archer API	Invalid or null token.	Each login account is limited to using five tokens at a time to download Archer reports. The last five of generated tokens are valid. All tokens expire after 30 days whether used or not. Using invalid tokens returns HTTP status code 403.
Archer API	Incorrect date format.	The date format used by the <code>date_from</code> and <code>date_to</code> fields is dd-mm-yyyy, meaning, date then month and year.
Archer API	The report not ready.	If you try to call <code>downloadarcherreport.do</code> before <code>generatearcherreport.do</code> has completed, you receive HTTP status code 204 to indicate no content is available. Try to download the report at a later time. After an excessively long time, if the Veracode Platform does not return the report, contact Veracode Technical Support.
Archer API	The results file is too large.	<p>When attempting to fetch the Archer feed for a large number of applications at once, the Veracode Platform may return HTTP status code 500. It is best in these cases to fetch the data using the optional arguments for the Archer API to limit the scope of the data being pulled (e.g. using <code>scan_type</code> and/or a date range). Once all the historical data is in place, use one of the <code>period</code> arguments (<code>yesterday</code>, <code>last_week</code>, or <code>last_month</code>) to pull data on a scheduled basis.</p> <p>Alternatively, you can use the asynchronous calls, downloadarcherreport.html and generatearcherreport.html.</p>
Eclipse Plugin	I experience a PKIX path building failure when installing the plugin from Eclipse.	<p>Add the following lines to the <code>eclipse.ini</code> file in your Eclipse installation directory:</p> <pre> -vmargs -- Djavax.net.ssl.trustStore="path for cacerts" -- Djavax.net.ssl.trustAnchors="path for cacerts" </pre>

API	Issue	Solution
Jenkins Plugin	<p>I receive one of these messages:</p> <ul style="list-style-type: none"> An <code>app_id</code> could not be located for application profile Access denied 	<ul style="list-style-type: none"> Check the Veracode user role for the logged-in account to verify that you have a role with permissions to create an application profile, such as Upload API for non-human API users or Creator for human users. Confirm that the Veracode application profile for the specified application name is visible by the specific teams who have access to this application and its scan results.
Jenkins Plugin	<p>The following message appears in the console output: <code>The policy status 'Did Not Pass' is not passing. Unable to continue.</code></p>	<p>This message indicates that you selected the Wait for scan to complete checkbox in your job configuration and the scan failed to pass your policy. If you want builds for scans that fail policy to complete, you must deselect that checkbox.</p>
Jenkins Plugin	<p>The test connection action fails. There is no success message.</p>	<ul style="list-style-type: none"> Verify that your Jenkins server has Internet connectivity. Check outside of the Jenkins plugin environment to verify if the server the Jenkins tool is running on has internet connectivity. To determine connectivity, download and run the Veracode Java API wrapper on the same machine the Jenkins tools are running on to test for internet connectivity. Verify the proxy settings to see if a proxy is required. If a proxy is not required, you can test for an external Internet connection with a cURL command and running, for example, the getapplist.do command.
Jenkins Plugin or Java API Wrapper	<p>The following message appears: <code>Requested array size exceeds VM limit.</code></p>	<p>This error indicates you are attempting to upload an archive that is too large for the current limit (in GB). Check the content and size of the files or archives you are uploading to verify you are using the correct files.</p>
Jenkins Plugin or Java	<p>The following message appears: <code>[16.01.11 14:28:39] java/net/HttpURLConnection.setFixedLengthStreamingMode(J)V Build step Upload and Scan with Veracode</code></p>	<p>This message indicates that the Java version you are using is not Java 7 or later. The Veracode Jenkins Plugin and the Veracode Java API wrapper require Java 7 or later.</p>

API	Issue	Solution
API Wrapper	marked build as failure Finished: FAILURE	
JIRA Plugin	If you need to troubleshoot any issues, turn debug logging on. You can review the logs at <JIRA home directory>/Application Data/JIRA/log/atlassian-jira.log.	<ol style="list-style-type: none"> 1 Stop JIRA, ensuring it is no longer running. 2 Open <code>atlassian-jira/WEB-INF/classes/log4j.properties</code> and change <code>log4j.logger.com.veracode.jira.plugin.synchronize = INFO,</code> <code>console, filelog</code> to <code>log4j.logger.com.veracode.jira.plugin.synchronize = DEBUG,</code> <code>console, filelog</code> 3 Restart JIRA. <p>When you have completed any debugging, be sure to change the DEBUG parameter back to INFO and restart JIRA.</p>
Maven Build Script	The following message is returned: <code>java.lang.ClassNotFoundException: Cannot find the specified class com.ibm.websphere.ssl</code>	The IBM WebSphere environment may prevent a Veracode UploadandScan target from executing if the Maven build script dependencies with the Java class path are missing. To resolve this, generate two <code>pom.xml</code> scripts, using one specifically for the Veracode upload.
Results API	The <code>getappbuilds.do</code> call is slow to deliver information.	Veracode recommends that you use <code>getapplist.do</code> to generate a list of all applications and <code>getbuildlist.do</code> to generate a list of all builds for an application. You can then use getappinfo.do and getbuildinfo.do to retrieve the information about specific applications and builds.
Upload API	I do not know if the prescan is complete or successful.	To check the prescan results in the Upload API, call getprescanresults.do .
Upload API	My scan does not complete due to non-fatal errors.	If you want to ensure the scan completes even though there are non-fatal errors such as unsupported frameworks, ensure you use the <code>scan_all_top_level_modules</code> parameter when you use the beginscan.do call.
Upload API and Integrations	I received a fatal error after prescan, which is preventing my static analysis from starting automatically.	Before the next time your static analysis is scheduled to start automatically, you need to:

API	Issue	Solution
		<ol style="list-style-type: none"> 1 Review the prescan results to identify the modules that have fatal errors. 2 Resolve the errors. <p>Optionally, if you do not want to resolve the errors, you can:</p> <ul style="list-style-type: none"> • Update your uploaded files to remove the modules that have errors. • Start the analysis manually. <p>If you have not added or deleted any modules since the last analysis that contained the fatal errors, the next automated analysis uses the same selected modules.</p>
Any Plugin, Any API	When using either a Veracode plugin, the Veracode API wrappers, or a custom script, I see this returned in the output text: <i>App not in state where new builds are allowed</i> .	This message indicates that a previous static scan did not succeed for the specific application. Log into the Veracode Platform and review the application's current scans to determine if the previous scan did not successfully complete. A previous scan may still be in progress. If a previous scan is still running due to an error, select Delete . You can then use the plugin to submit a new scan request.
Visual Studio Extension	I received a download error that says <i>No applications exist for the specified user's account</i> .	Using the Visual Studio Veracode menu, you may have attempted to download results after selecting a specific application for which you do not have permission to access. You must be a member of each team associated with an application to be able to access that application's data.
Visual Studio Extension	The Upload Build menu does not populate the Application dropdown list or allow me to complete the Build text box.	This message indicates that you do not have the required role to either create a new application or build.
Visual Studio Extension	I receive this message <i>Support Issue: No precompiled files were found for this ASP.NET web application</i> .	Use the Veracode Static for Visual Studio to prepare your .NET application for uploading to Veracode. For information on how to use this extension, see Using Veracode Static for Visual Studio .
Veracode Azure DevOps	The Veracode Release Summary report is not displaying in the TFS on-premise extension.	If you rename the build step task Upload and Scan, the extension cannot find and execute the task, and no Veracode Summary Report is created.

API	Issue	Solution
Extension		
Any IDE Plugin or Extension	I am having problems with my credentials and am prompted to enter them multiple times.	<ol style="list-style-type: none">1 Check that you have saved your credentials. Go to Veracode > Options > Credentials and verify the API ID and key credentials are saved.2 Verify that the field Do not use stored credentials to log in is not selected.

For assistance with errors you receive while compiling your application, see the [Troubleshooting Precompilation Errors](#) page.

If you cannot find the solution to your problem on this page, contact Veracode Technical Support.

REST APIs

Using the Veracode REST APIs

Veracode REST APIs enable you to access Veracode Platform data and functionality using normal REST API programming conventions.

To access the [REST APIs](#), you must use Veracode API [ID and key credentials](#) and [HMAC](#) to protect your Veracode account and keep your data secure.



NOTE: Not all Veracode functionality is provided via REST APIs, but may be covered by one of our [XML APIs](#).

AppSec APIs

Applications API

You can use the Veracode Applications API to quickly access information about your Veracode applications.

The Applications API endpoint provides access to all the applications in your portfolio, as well as application-related data, such as findings, sandboxes, and policy evaluations. You can use the Applications API with other REST APIs to simplify common reporting and dashboarding scenarios, enabling you to get the latest data for each application with a single call. You can also access historical state change information on findings and use HMAC authentication to improve API security.

Applications API Specification

The Applications API specification is available:

- [As a JSON file](#)
- [On SwaggerHub](#)

Policy API

The Veracode Policy API enables you to create, update, delete, and read policies and to evaluate an application or a sandbox against any policy.

The policy evaluation endpoint allows for assessing an application or sandbox against any policy, even one not currently assigned to the application. The response from the policy evaluation shows you why the application is passing or failing policy, including scan frequency requirements and findings that are past their grace period due date.

Policy API Specification

The Policy API specification is available:

- [As a JSON file](#)
- [On SwaggerHub](#)

Manual Testing API

The Veracode Manual Testing REST API provides access to details about published Veracode Manual Penetration Testing (MPT) scans and findings.

The Manual Testing findings endpoint works with the Findings API to provide more information about MPT findings, including detailed notes from the penetration tester, screenshots, and code samples, if provided.

Manual Testing API Specification

The Manual Testing API specification is available:

- [As a JSON file](#)
- [On SwaggerHub](#)

Use Case Scenarios

Create a Policy

Use this code to create a policy.

You must specify a policy definition.

The following example file `policy.json` specifies a policy definition.

```
{
  "name": "TestPolicy",
  "type": "CUSTOMER",
  "description": "Policy to test create endpoint in end to
end testing",
  "vendor_policy": false,
  "finding_rules": [
    {
      "type": "MAX_SEVERITY",
      "scan_type": [
        "DYNAMIC",
        "MANUAL",
        "STATIC"
      ],
      "value": "3"
    }
  ]
}
```

To create a policy using this sample file, run this command:

```
1 http --auth-type=veracode_hmac POST "https://api.veracode.com/
  appsec/v1/policies" < policy.json
```

Update a Policy

Use this code to update a policy.

Provide the updated values for an existing policy. The following example file `updated_policy.json` specifies a policy update.

```
{
```

```

    "name": "New TestPolicy",
    "type": "CUSTOMER",
    "description": "Policy to test create endpoint in end to end
testing",
    "vendor_policy": false,
    "finding_rules": [
        {
            "type": "MAX_SEVERITY",
            "scan_type": [
                "DYNAMIC",
                "MANUAL",
                "STATIC"
            ],
            "value": "4"
        }
    ]
}

```

To update a policy using this sample file, run this command:

```
1 http --auth-type=veracode_hmac PUT "https://api.veracode.com/
  appsec/v1/policies/{policy_guid}" < updated_policy.json"
```

Deleting a Policy

Use this code to delete a policy.

You can delete a policy using one of the following options:

- Provide a replacement policy for all applications for which you want to delete the policy. For example:

```
http --auth-type=veracode_hmac delete
  "https://api.veracode.com/appsec/v1/policies/{policy_guid}?
  replacement_GUID={replacement_policy_guid}"
```

- Enter the `delete` option without providing a replacement policy. In this scenario, specify a default policy instead of a replacement policy. For example:

```
http --auth-type=veracode_hmac delete
  "https://api.veracode.com/appsec/v1/policies/{policy_guid}?
  replace_with_default_policy=true"
```

Dynamic Analysis APIs

Dynamic Analysis API

Veracode Dynamic Analysis provides a REST API that fully automates the major dynamic scanning tasks. The ability to programmatically initiate dynamic scanning provides the flexibility necessary when incorporating dynamic scanning into your SDLC.

The API endpoints perform the following tasks:

- Create analyses with URL scans
- Configure analyses and URL scans

- Schedule and run analyses
- Link analyses to Veracode application profiles

Permissions and Authentication

To be able to use the Veracode Dynamic Analysis REST API, you must have either a:

- Non-human API user account with the [Upload and Scan API role](#)
- Human user account with the [Security Lead, Creator, or Submitter role](#)

The API provides improved security through HMAC authentication. Therefore, before using this API, you must first [configure your authentication](#).

Dynamic Analysis API Specification

The Dynamic Analysis API specification is available:

- [As a YAML file](#)
- [On SwaggerHub](#)

Use Case Scenarios

Create a Dynamic Analysis of a Single URL

Use this code to submit a simple Dynamic Analysis of a single URL.

To create an analysis of a URL, make a POST call to `https://api.veracode.com/was/configservice/v1/analyses`, specifying the timeframe number, unit, and schedule parameters.

The following is an example of a Dynamic Analysis for one URL that runs immediately for one day.

```
{
  "name": "Veracode API Analysis",
  "scans": [
    {
      "scan_config_request": {
        "target_url": {
          "url": "http://www.example.com"
        }
      }
    }
  ],
  "schedule": {
    "duration": {
      "length": 1,
      "unit": "DAY"
    },
    "scheduled": true,
    "now": true
  }
}
```

Create a Dynamic Analysis of Multiple URLs

Use this code to create a Dynamic Analysis to scan multiple URLs with visibility, organization, and schedule restrictions.

To create an analysis of multiple URLs:

- 1 Log in to the Veracode Platform to obtain the ID of the business unit that owns the Dynamic Analysis.
- 2 Make a POST call to `https://api.veracode.com/was/configservice/v1/analyses`, specifying the schedule parameters, business ownership, and visibility settings.

The following is an example of a Dynamic Analysis for multiple URLs. This analysis runs on a specific day for three days, is owned by business unit 12345, and is only visible to users in the business unit who have the Security Lead [role](#).

```
{
  "name": "Veracode API Analysis",
  "scans": [
    {
      "scan_config_request": {
        "target_url": {
          "url": "https://www.example.com/one/"
        }
      }
    },
    {
      "scan_config_request": {
        "target_url": {
          "url": "https://www.example.com/two/"
        }
      }
    }
  ],
  "schedule": {
    "start_date": "2020-09-26T02:00+00:00",
    "duration": {
      "length": 3,
      "unit": "DAY"
    }
  },
  "visibility": {
    "setup_type": "SEC_LEADS_ONLY",
    "team_identifiers": []
  },
  "org_info": {
    "business_unit_id": "12345",
    "owner": "Test Development Team",
    "email": "user@example.com"
  }
}
```

Create a Dynamic Analysis with Authentication

The Dynamic Analysis API enables you to use auto-login, basic authentication, and form-based login with a login script.

To create an analysis of multiple URLs with authentication, make a POST call to `https://api.veracode.com/was/configservice/v1/analyses`, specifying the authentication details.

The following are examples of the different types of authentication you can use with a Dynamic Analysis:

Auto-login

```
{
  "name": "Veracode API Scan test Auto-Login",
  "scans": [
    {
      "scan_config_request": {
        "target_url": {
          "url": "http://www.example.com",
          "http_and_https": true,
          "directory_restriction_type":
"DIRECTORY_AND_SUBDIRECTORY"
        },
        "auth_configuration": {
          "authentications": {
            "AUTO": {
              "username": "user",
              "password": "pass",
              "authtype": "AUTO"
            }
          }
        }
      }
    }
  ],
  "schedule": {
    "now": true,
    "duration": {
      "length": 1,
      "unit": "DAY"
    }
  }
}
```

Basic Authentication

```
{
  "name": "Veracode API BASIC Auth",
  "scans": [
    {
      "scan_config_request": {
        "target_url": {
          "url": "http://www.example.com",
          "http_and_https": true,
          "directory_restriction_type":
"DIRECTORY_AND_SUBDIRECTORY"
        },
        "auth_configuration": {
          "authentications": {
            "BASIC": {
```

```

        "username": "username",
        "password": "pass",
        "authtype": "BASIC"
    }
}
},
"schedule": {
    "now": true,
    "duration": {
        "length": 1,
        "unit": "DAY"
    }
}
}
}

```

Form-Based Login with Login Script



NOTE: Before you paste the crawl script into the API body, you must escape the JSON. If you need assistance, use the JSON escape utility available at <https://jsonformatter.org>.

```

{
  "name": "Veracode Form Based login test API",
  "scans": [
    {
      "scan_config_request": {
        "target_url": {
          "url": "http://www.example.com",
          "http_and_https": true,
          "directory_restriction_type":
"DIRECTORY_AND_SUBDIRECTORY"
        },
        "authentications": {
          "FORM": {
            "script_file": "example-login-script.side",
            "login_script_data": {
              "script_body": "<escaped JSON>",
              "script_type": "SELENIUM"
            },
            "authtype": "FORM"
          }
        }
      }
    }
  ],
  "schedule": {
    "now": true,
    "duration": {
      "length": 1,
      "unit": "DAY"
    }
  }
}

```


Create a Dynamic Analysis with a Crawl Script

The Dynamic Analysis API enables you to use a prerecorded crawl sequence to supplement the crawl automation of the Veracode scan engine.

To create an analysis that includes a crawl script, make a POST call to `https://api.veracode.com/was/configservice/v1/analyses`, specifying the authentication details.

The following is an example of how to use a Dynamic Analysis with a crawl script:



NOTE: Before you paste the crawl script into the API body, you must escape the JSON. If you need assistance, use the JSON escape utility available at <https://jsonformatter.org>.

```
{
  "name": "Veracode API",
  "scans": [
    {
      "scan_config_request": {
        "target_url": {
          "url": "http://www.example.com",
          "http_and_https": true,
          "directory_restriction_type":
"DIRECTORY_AND_SUBDIRECTORY"
        },
        "crawl_configuration": {
          "scripts": [
            {
              "crawl_script_data": {
                "script_type": "SELENIUM",
                "script_body": "<escaped JSON>"
              },
              "name": "ExampleCrawlScript.side"
            }
          ],
          "disabled": false
        }
      }
    },
    {
      "schedule": {
        "now": true,
        "duration": {
          "length": 1,
          "unit": "DAY"
        }
      }
    }
  ]
}
```

Create a Dynamic Analysis of a Single URL with Prescan Verification

Use this code to submit a Dynamic Analysis with prescan verification.

To be able to run a prescan verification before the Dynamic Analysis, you must allow two hours for the prescan to complete before you schedule the analysis to begin.

To create an analysis of a URL with prescan verification, make a POST call to `https://api.veracode.com/was/configservice/v1/analyses?run_verification=true`.

If your prescan verification fails, the Dynamic Analysis also fails.

You can call `GET /analyses` to see the status of your analyses.

Create a Dynamic Analysis with a Recurring Schedule

Use this code to create a Dynamic Analysis that repeats on a recurring basis.

To create an analysis with a recurring schedule, make a POST call to `https://api.veracode.com/was/configservice/v1/analyses`, specifying the schedule parameters and the recurrence configuration.

The following is an example of the code to add to your call when you want to make your schedule a recurring event.

```
{
  "name": "Veracode API Analysis",
  "scans": [
    {
      "scan_config_request": {
        "target_url": {
          "url": "http://www.example.com"
        }
      }
    }
  ],
  "schedule": {
    "now": true,
    "duration": {
      "length": 1,
      "unit": "DAY"
    },
    "end_date": "",
    "scan_recurrence_schedule": {
      "recurrence_type": "WEEKLY",
      "schedule_end_after": 2,
      "recurrence_interval": 1,
      "day_of_week": "FRIDAY"
    }
  }
}
```

Create a Dynamic Analysis with a Pause and Resume Schedule

Use this code to create a Dynamic Analysis with a pause and resume schedule.

To create an analysis with a pause and resume schedule, make a POST call to `https://api.veracode.com/was/configservice/v1/analyses`, specifying the schedule parameters, including the blackout timeframe.

The following is an example of a Dynamic Analysis that contains a pause and resume timeframe between 9:00pm and 11:00pm.

```
{
  "name": "Veracode API Analysis",
  "scans": [
    {
      "scan_config_request": {
        "target_url": {
          "url": "http://www.example.com"
        }
      }
    }
  ],
  "schedule": {
    "start_date": "2019-09-27T16:49:00-04:00",
    "duration": {
      "length": 1,
      "unit": "DAY"
    }
  },
  "scan_blackout_schedule": {
    "blackout_type": "THESE_HOURS",
    "blackout_start_time": "21:00",
    "blackout_end_time": "23:00",
    "blackout_days": null
  }
}
```

Create a Dynamic Analysis of Multiple URLs with User Agent and Blacklist

Use this code to create a Dynamic Analysis to scan multiple URLs, specify a user agent, and identify blacklisted URLs.

To create a Dynamic Analysis of multiple URLs with user agent and blacklist configurations, make a POST call to <https://api.veracode.com/was/configservice/v1/analyses>, specifying the user agent details and any URLs you do not want Veracode to scan.

If you want to specify a custom header, the type field must be CUSTOM. If you use a predefined agent, such as IE11, you cannot include a custom_header field.

The following is an example of a Dynamic Analysis of multiple URLs. This analysis specifies the agent user string and the URLs to exclude from both the analysis and the URL scan.



NOTE: The URLs that you blacklist at the URL-specific level take precedence over the URLs in this blacklist. For example, if you blacklist a URL in your URL configuration, but not in the configuration of the whole Dynamic Analysis, the URL configuration blacklist determines which URLs to exclude from the scan.

```
{
  "name": "Veracode API ",
```

```

"scans": [
  {
    "scan_config_request": {
      "target_url": {
        "url": "http://www.example.com/one/"
      },
      "scan_setting": {
        "user_agent": {
          "type": "CUSTOM",
          "custom_header": "Custom User Agent String"
        },
        "blacklist_configuration": {
          "blackList": [
            {
              "url": "http://www.example.com/one/
black/",
              "http_and_https": true
            }
          ]
        }
      },
    },
    {
      "scan_config_request": {
        "target_url": {
          "url": "http://www.example.com/two/",
          "http_and_https": true
        }
      }
    }
  ],
  "org_info": {
    "email": "user@example.com"
  },
  "visibility": {
    "setup_type": "SEC_LEADS_ONLY",
    "team_identifiers": []
  },
  "scan_setting": {
    "blacklist_configuration": {
      "blackList": [
        {
          "url": "http://www.example.com/black1/",
          "http_and_https": false
        },
        {
          "url": "http://www.example.com/black2/
site.html",
          "http_and_https": false
        }
      ]
    },
    "user_agent": {
      "type": "CUSTOM",
      "custom_header": "Mozilla/5.0 (Windows NT 6.1;
WOW64; Trident/7.0; rv:11.0) like Gecko VERACODE"
    }
  }
}

```

Link Dynamic Analysis Results to an Application

Use this code to link the results of a Dynamic Analysis to an existing application.

To link results to an application:

- 1 Make a GET call to `https://api.veracode.com/was/configservice/v1/platform_applications` to find your application by name and obtain the UUID.
- 2 Make a POST call to `https://api.veracode.com/was/configservice/v1/analyses` to create an analysis, specifying the ID of the application to which you want to link the analysis results.

The following is an example of how to link scan results from the specified target URL to the UUID of an existing application.

```
{
  "name": "Veracode API applink",
  "scans": [
    {
      "linked_platform_app_uuid": "abcd1234-e6d0-475d-ac70-abff5388fa75",
      "scan_config_request": {
        "target_url": {
          "url": "http://www.example.com/",
          "http_and_https": true,
          "directory_restriction_type": "DIRECTORY_AND_SUBDIRECTORY"
        }
      }
    }
  ],
  "schedule": {
    "now": true,
    "duration": {
      "length": 1,
      "unit": "DAY"
    }
  }
}
```

Configure a Dynamic Analysis for Internal Scanning

Use this code to configure a Dynamic Analysis to run with Veracode Internal Scanning Management (ISM).

To configure an analysis to run with ISM:

- 1 Go to `https://ui.analysiscenter.veracode.com/mvsa/gms/admin/gateways?depth=1`.
- 2 Obtain the:
 - Gateway ID by looking for the `ref_id` corresponding to the gateway name
 - Endpoint ID by looking for the `token` corresponding to the gateway name
- 3 Make a POST call to `https://api.veracode.com/was/configservice/v1/analyses` to create an analysis, specifying the endpoint and gateway IDs.

The following is an example of a Dynamic Analysis configured for ISM.

```
{
  "name": "Veracode API ISM",
  "scans": [
    {
      "scan_config_request": {
        "target_url": {
          "url": "http://www.example.com/",
          "http_and_https": true,
          "directory_restriction_type":
            "DIRECTORY_AND_SUBDIRECTORY"
        }
      },
      "internal_scan_configuration": {
        "enabled": true,
        "gateway_id": "12345678-85b2-4fe2-
a633-425418f6a5ef",
        "endpoint_id": "1234abcd-cecc-43e8-
a658-6b3447c3e52a"
      }
    }
  ],
  "schedule": {
    "now": true,
    "duration": {
      "length": 1,
      "unit": "DAY"
    }
  }
}
```

Edit a Dynamic Analysis to Add a URL Scan

Use this code to edit an existing Dynamic Analysis to add an additional URL scan.

To edit an existing analysis:

- 1 Find the `analysis_id` of the analysis you want to edit using `https://api.veracode.com/was/configservice/v1/analyses?name={name}`, where `name` is the name of the analysis you want to edit.
- 2 Add the additional URL you want to scan in a PUT call to `https://api.veracode.com/was/configservice/v1/analyses/{analysis_id}?method=PATCH`.

The following is an example of the payload you would enter in the PUT call to edit an existing analysis.

```
{
  "scans": [
    {
      "scan_config_request": {
        "target_url": {
          "url": "https://www.example.com"
        }
      }
    }
  ]
}
```

```

    },
    "action_type": "ADD"
  }
]
}

```

Edit a Dynamic Analysis to Delete a URL Scan

Use this code to edit a Dynamic Analysis to edit a URL scan from the analysis.

To edit an existing analysis:

- 1 Find the `analysis_id` of the analysis you want to edit using `https://api.veracode.com/was/configservice/v1/analyses?name={name}`, where `name` is the name of the analysis you want to edit.
- 2 Find the `scan_id` using `https://api.veracode.com/was/configservice/v1/analyses/{analysis_id}/scans`.
- 3 Specify the `scan_id` of the scan you want to remove in a PUT call to `https://api.veracode.com/was/configservice/v1/analyses/{analysis_id}?method=PATCH`

The following is an example of the payload you would put in a PUT call to delete a URL scan from an analysis.

```

{
  "scans": [
    {
      "action_type": "REMOVE",
      "scan_id": "<scan_id>"
    }
  ]
}

```

Edit the Schedule of a Dynamic Analysis

Use this code to edit the schedule of a Dynamic Analysis.

To edit an existing schedule of an analysis:

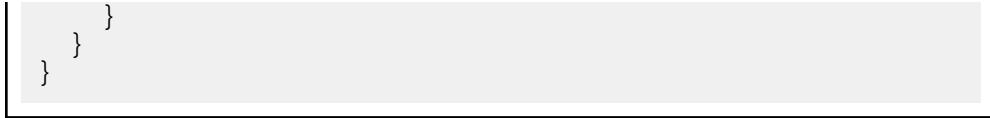
- 1 Find the `analysis_id` of the analysis you want to edit using `https://api.veracode.com/was/configservice/v1/analyses?name={name}`, where `name` is the name of the analysis you want to edit.
- 2 Put the new schedule information in a PUT call to: `https://api.veracode.com/was/configservice/v1/analyses/{analysis_id}?method=PATCH`

The following is an example of the code you would put in a PUT call to change the schedule of an analysis.

```

{
  "schedule": {
    "start_date": "2020-09-26T02:00+00:00",
    "duration": {
      "length": 3,
      "unit": "DAY"
    }
  }
}

```



eLearning APIs

Using the Veracode eLearning APIs

The Veracode eLearning APIs enable you to view and manage the courses, users, and user progress associated with the Veracode eLearning accounts in your organization.

Veracode eLearning uses REST API calls. To use the Veracode eLearning APIs, you must have valid [Veracode API credentials](#) and [enable HMAC authentication within your application](#) as a security measure for accessing API resources.

Veracode eLearning Courses Available API

Use the Veracode eLearning Courses Available API to return a collection of Veracode eLearning courses that are available for your organization.

Veracode eLearning Course by ID API

Use the Veracode eLearning Course by ID API to return a name of a course for the specified ID.

Veracode eLearning Users API

Use the Veracode eLearning Users API to return a collection of users you manage. If you are a user with the Executive and Veracode eLearning role, the response includes all Veracode eLearning users in your organization.

Veracode eLearning User by ID API

Use the eLearner by ID API to return a name of a user for the specified user ID.

Veracode eLearning Progress API

Use the Veracode eLearning Progress API to return a collection of progress status report cards for Veracode eLearning users you manage. If you are a user with the Executive and Veracode eLearning role, the response includes report cards of all Veracode eLearning users of the organization.

Veracode eLearning Courses Available API

Use the Courses Available API to return a collection of Veracode eLearning courses in JSON format that are available for your organization.

Resource URL

<https://api.veracode.com/elearning/v1/courses>

Table 2: Parameters

Name	Query or Path Parameter	Type	Description
course_id	Query	String	Optional. Course identifiers. Repeat the query parameters to specify multiple values.

Name	Query or Path Parameter	Type	Description
page	Query	Integer	Optional. Page number. Default is 0.
size	Query	Integer	Optional. Page size. Default is 50 courses per page, maximum is 500 courses per page.

Table 3: Response Codes

HTTP Code	Description	Type
200	Success. Returns an array of Veracode eLearning courses.	Object
400	Invalid request.	Null
404	Not found.	Null
403	Access denied.	Null
500	Server-side error.	Null

Example Request

The following example assumes you have correctly configured your credentials and configured any required HMAC authentication libraries.

```
import java.net.URI;

import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import org.json.JSONObject;

import com.veracode.security.apisigning.ClientCryptoLib;

public class GetCoursesClient {

    // An API Id for authentication
    private final static String API_KEY = "API_KEY_GOES_HERE";

    // The secret key corresponding to the API Id
    private final static String API_SECRET = "API_SECRET_GOES_HERE";

    public static void main(String[] args) throws Exception {

        URI uri = URI.create("https://api.veracode.com/elearning/v1/courses");
        String authHeader =
            ClientCryptoLib.calculateAuthorizationHeader(ClientCryptoLib.VERACODE_
                HMAC_SHA_256, API_KEY,
                API_SECRET, uri.getHost(), uri.getPath(),
```

```

HttpGet.METHOD_NAME);

    HttpGet request = new HttpGet(uri);
    request.addHeader("Authorization", authHeader);

    CloseableHttpResponse response =
HttpClients.createDefault().execute(request);
    HttpEntity entity = response.getEntity();
    JSONObject json = new JSONObject(EntityUtils.toString(entity,
"UTF-8"));

    System.out.println(json.toString(4));
}
}

```

Example Response

The following response conforms to the Hypertext Application Language content type (application/hal+json), which includes a link to the `reportcards` endpoint for the courses.

```

GET/elearning/v1/coursesHTTP/1.1
    HTTP/1.1200OK,Content-Type:application/json
{
  "_embedded": {"courses": [
    {
      "_links": {
        "reportcards": {
          "templated": true,
          "href": "https://api.veracode.com/elearning/v1/
reportcards?course_id=CRLF{&user_id,page,size}"
        },
        "self": {"href": "https://api.veracode.com/
elearning/v1/courses/CRLF"}
      },
      "name": "AppSec Tutorials - CRLF Injection",
      "courseId": "CRLF"
    },
    { ... }
  ]},
  "_links": {
    "next": {"href": "https://api.veracode.com/elearning/v1/
courses?page=1&size=50"},
    "last": {"href": "https://api.veracode.com/elearning/v1/
courses?page=1&size=50"},
    "self": {
      "templated": true,
      "href": "https://api.veracode.com/elearning/v1/courses?
page=0&size=50{&course_id}"
    },
    "first": {"href": "https://api.veracode.com/elearning/v1/
courses?page=0&size=50"}
  },
  "page": {
    "number": 0,
    "size": 50,
    "totalPages": 2,
    "totalElements": 54
  }
}

```

```
}
}
```

Veracode eLearning Course by ID API

Use the Veracode eLearning Course by ID API to return a name of a course in JSON format for the specified ID. For the list of IDs for the courses available to your organization, use the [Courses Available API](#).

Resource URL

https://api.veracode.com/elearning/v1/courses/{course_id}

Table 4: Parameters

Name	Query or Path Parameter	Type	Description
course_id * (required)	Path	String	The course unique identifier.

Table 5: Response Codes

HTTP Code	Description	Type
200	Success. Returns a course for the specified ID.	Course
400	Invalid request.	Null
404	Not found.	Null
403	Access denied.	Null
500	Server-side error.	Null

Example Request

The following example assumes you have correctly configured your credentials and configured any required HMAC authentication libraries.

```
import java.net.URI;

import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import org.json.JSONObject;

import com.veracode.security.apisigning.ClientCryptoLib;

public class GetCourseClient {

    // An API Id for authentication
    private final static String API_KEY = "API_KEY_GOES_HERE";

    // The secret key corresponding to the API Id
```

```

private final static String API_SECRET = "API_SECRET_GOES_HERE";

public static void main(String[] args) throws Exception {

    URI uri = URI.create("https://api.veracode.com/elearning/v1/
courses/CRLF");
    String authHeader =
ClientCryptoLib.calculateAuthorizationHeader(ClientCryptoLib.VERACODE_
HMAC_SHA_256, API_KEY,
        API_SECRET, uri.getHost(), uri.getPath(),
HttpGet.METHOD_NAME);

    HttpGet request = new HttpGet(uri);
    request.addHeader("Authorization", authHeader);

    CloseableHttpResponse response =
HttpClient.createDefault().execute(request);
    HttpEntity entity = response.getEntity();
    JSONObject json = new JSONObject(EntityUtils.toString(entity,
"UTF-8"));

    System.out.println(json.toString(4));
}
}

```

Example Response

The following response conforms to the Hypertext Application Language content type (application/hal+json), which includes a link to the reportcards endpoint for the given course.

```

GET/elearning/v1/courses/{course_id}HTTP/1.1
HTTP/1.1200OKContent-Type:application/json
{
  "_links": {
    "reportcards": {
      "templated": true,
      "href": "https://api.veracode.com/elearning/v1/
reportcards?course_id=CRLF{&user_id,page,size}"
    },
    "self": {"href": "https://api.veracode.com/elearning/v1/
courses/CRLF"}
  },
  "name": "AppSec Tutorials - CRLF Injection",
  "courseId": "CRLF"
}

```

Veracode eLearning Users API

Use the Veracode eLearning Users API to return a list of IDs for the Veracode eLearning users in your organization.

Resource URL

<https://api.veracode.com/elearning/v1/learners>

Table 6: Parameters

Name	Query or Path Parameter	Type	Description
user_id	Query	String	Optional. User identifiers. Repeat query parameters to specify multiple values.
page	Query	Integer	Optional. Page number. Default is 0.
size	Query	Integer	Optional. Page size. Default is 50 users per page, maximum is 500 users per page.

Table 7: Response Codes

HTTP Code	Description	Type
200	Success. Returns an array of eLearning users.	Object
400	Invalid request.	Null
404	Not found.	Null
403	Access denied.	Null
500	Server-side error.	Null

Example Request

The following example assumes you have correctly configured your credentials and configured any required HMAC authentication libraries.

```
import java.net.URI;

import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import org.json.JSONObject;

import com.veracode.security.apisigning.ClientCryptoLib;

public class GetLearnersClient {

    // An API Id for authentication
    private final static String API_KEY = "API_KEY_GOES_HERE";

    // The secret key corresponding to the API Id
    private final static String API_SECRET = "API_SECRET_GOES_HERE";
```

```

    public static void main(String[] args) throws Exception {
        URI uri = URI.create("https://api.veracode.com/elearning/v1/learners");
        String authHeader =
            ClientCryptoLib.calculateAuthorizationHeader(ClientCryptoLib.VERACODE_
                HMAC_SHA_256, API_KEY,
                API_SECRET, uri.getHost(), uri.getPath(),
                HttpGet.METHOD_NAME);

        HttpGet request = new HttpGet(uri);
        request.addHeader("Authorization", authHeader);

        CloseableHttpResponse response =
            HttpClientBuilder.create().execute(request);
        HttpEntity entity = response.getEntity();
        JSONObject json = new JSONObject(EntityUtils.toString(entity,
            "UTF-8"));

        System.out.println(json.toString(4));
    }
}

```

Example Response

The following response conforms to the Hypertext Application Language content type (application/hal+json), which includes a link to the `reportcards` endpoint for the given user.

```

GET/elearning/v1/learnersHTTP/1.1HTTP/1.1
200OKContent-Type:application/json
{
  "_embedded": {"learners": [{
    "firstName": "John",
    "lastName": "Smith",
    "lastLogin": "2018-05-11T20:05:55.030Z",
    "_embedded": {"curricula": [{"curriculumName": "AllReqd"}]},
    "_links": {
      "reportcards": {
        "templated": true,
        "href": "https://api.veracode.com/elearning/v1/
reportcards?user_id=jsmith{&course_id,page,size}"
      },
      "self": {"href": "https://api.veracode.com/elearning/v1/
learners/jsmith"}
    },
    "userId": "jsmith",
    "email": "jsmith@example.com"
  }]},
  "_links": {"self": {
    "templated": true,
    "href": "https://api.veracode.com/elearning/v1/learners?
page=0&size=50{&user_id}"
  }},
  "page": {
    "number": 0,
    "size": 50,
    "totalPages": 1,
    "totalElements": 1
  }
}

```

```
}
}
```

eLearner by ID API

Use the eLearner by ID API to return user name data in JSON format for the specified user ID. For the list of IDs for the users in your organization, use the [eLearning Users API](#).

Resource URL

https://api.veracode.com/elearning/v1/learners/{user_id}

Table 8: Parameters

Name	Query or Path Parameter	Type	Description
user_id * (required)	Path	String	The eLearning user unique identifier.

Table 9: Response Codes

HTTP Code	Description	Type
200	Success. Returns an eLearning user for the specified ID.	Learner
400	Invalid request.	Null
401	Unauthorized request.	Null
403	Access denied.	Null
500	Server side error.	Null

Example Request

The following example assumes you have correctly configured your credentials and configured any required HMAC authentication libraries.

```
import java.net.URI;

import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import org.json.JSONObject;

import com.veracode.security.apisigning.ClientCryptoLib;

public class GetLearnerClient {

    // An API Id for authentication
    private final static String API_KEY = "API_KEY_GOES_HERE";

    // The secret key corresponding to the API Id
```

```

private final static String API_SECRET = "API_SECRET_GOES_HERE";

public static void main(String[] args) throws Exception {

    URI uri = URI.create("https://api.veracode.com/elearning/v1/
learners/jsmith");
    String authHeader =
ClientCryptoLib.calculateAuthorizationHeader(ClientCryptoLib.VERACODE_
HMAC_SHA_256, API_KEY,
        API_SECRET, uri.getHost(), uri.getPath(),
HttpGet.METHOD_NAME);

    HttpGet request = new HttpGet(uri);
    request.addHeader("Authorization", authHeader);

    CloseableHttpResponse response =
HttpClient.createDefault().execute(request);
    HttpEntity entity = response.getEntity();
    JSONObject json = new JSONObject(EntityUtils.toString(entity,
"UTF-8"));

    System.out.println(json.toString(4));
}
}

```

Example Response

The following response conforms to the Hypertext Application Language content type (application/hal+json), which includes a link to the `reportcards` endpoint for the given user.

```

GET/elearning/v1/learners/{user_id}HTTP/1.1
HTTP/1.1200OKContent-Type:application/json
{
  "firstName": "John",
  "lastName": "Smith",
  "lastLogin": "2018-05-11T19:23:28.027Z",
  "_embedded": {"curricula": [{"curriculumName": "AllReqd"}]},
  "_links": {
    "reportcards": {
      "templated": true,
      "href": "https://api.veracode.com/elearning/v1/reportcards?
user_id=jsmith{&course_id,page,size}"
    },
    "self": {"href": "https://api.veracode.com/elearning/v1/
learners/jsmith"}
  },
  "userId": "jsmith",
  "email": "jsmith@example.com"
}

```

Veracode eLearning Progress API

Use the Veracode eLearning Progress API to return a collection of progress status report cards in JSON format for Veracode eLearning users you manage. If you have the Executive and eLearning roles, the response includes report cards of all Veracode eLearning users of the organization.

Resource URL

<https://api.veracode.com/elearning/v1/reportcards>

Table 10: Parameters

Name	Query or Path Parameter	Type	Description
user_id	Query	String	Optional. User identifier.
course_id	Query	String	Optional. Course identifier.
page	Query	Integer	Optional. Page number. Default is 0.
size	Query	Integer	Optional. Page size. Default is 50 users per page, maximum is 500 users per page.

Table 11: Response Codes

HTTP Code	Description	Type
200	An array of progress status report cards.	Object
400	Invalid request.	Null
404	Not found.	Null
403	Access denied.	Null
500	Server-side error.	Null

Example Request

The following example assumes you have correctly configured your credentials and configured any required HMAC authentication libraries.

```
import java.net.URI;

import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import org.json.JSONObject;

import com.veracode.security.apisigning.ClientCryptoLib;

public class GetReportCardsClient {

    // An API Id for authentication
    private final static String API_KEY = "API_KEY_GOES_HERE";
```

```
// The secret key corresponding to the API Id
private final static String API_SECRET = "API_SECRET_GOES_HERE";

public static void main(String[] args) throws Exception {

    URI uri = URI.create(
        "https://api.veracode.com/elearning/v1/reportcards?
course_id=CRLF&user_id=jsmith");
    String authHeader =
        ClientCryptoLib.calculateAuthorizationHeader(ClientCryptoLib.VERACODE_
        HMAC_SHA_256, API_KEY,
        API_SECRET, uri.getHost(),
        uri.getPath().concat("?" + uri.getQuery()), HttpGet.METHOD_NAME);

    HttpGet request = new HttpGet(uri);
    request.addHeader("Authorization", authHeader);

    CloseableHttpResponse response =
        HttpClient.createDefault().execute(request);
    HttpEntity entity = response.getEntity();
    JSONObject json = new JSONObject(EntityUtils.toString(entity,
        "UTF-8"));

    System.out.println(json.toString(4));
}
}
```

Example Response

The following response conforms to the Hypertext Application Language content type (application/hal+json), which includes a link to the learner endpoint and course endpoint.

```
GET/elearning/v1/reportcardsHTTP/1.1
HTTP/1.1200OKContent-Type:application/json
{
  "_embedded": {"reportcards": [{
    "timeSpentOnCourse": 0.24,
    "_links": {
      "learner": {"href": "https://api.veracode.com/elearning/v1/
learners/jsmith"},
      "course": {"href": "https://api.veracode.com/elearning/v1/
courses/CRLF"}
    },
    "courseStartedDate": "2018-05-11T00:00:00.000Z",
    "courseStatus": "Passed",
    "progressPercent": 100,
    "numberOfAttempts": 1,
    "courseCompletedDate": "2018-05-11T00:00:00.000Z"
  ]}],
  "_links": {"self": {"href": "https://api.veracode.com/elearning/v1/
reportcards?user_id=jsmith&course_id=CRLF&page=0&size=50"}},
  "page": {
    "number": 0,
    "size": 50,
    "totalPages": 1,
    "totalElements": 1
  }
}
```

```
}  
}
```

Greenlight APIs

Greenlight API

The Veracode Greenlight API enables you to start a Greenlight Java binary scan or obtain results from previous scans, directly in the CI pipeline outside an IDE. The endpoint provides improved security through HMAC authentication. Therefore, before using this API, you must first [configure your authentication](#).

Greenlight API Specification

The Greenlight API specification is available:

- [As a JSON file](#)
- [On SwaggerHub](#)

Software Composition Analysis APIs

Veracode SCA Agent API

You can use the Veracode SCA Agent REST API to programmatically extract high-level workspace information on specific Veracode Agent-Based Scan workspaces or all workspaces to which you have access. You can also filter your workspaces on library, vulnerability, and license.



NOTE: To prevent your API requests being blocked, provide your company, application name, or email domain in your User-Agent header so that Veracode Technical Support can address any issues.

The Veracode SCA Agent REST API requires an Enterprise subscription plan. To get a quote for an upgrade to Enterprise, send an email to contact@veracode.com.

Veracode SCA Agent API Specification

The Veracode SCA Agent API specification is available:

- [As a JSON file](#)
- [On SwaggerHub](#)



NOTE: There are two host URLs for the Veracode SCA Agent API. If you use SourceClear API credentials, the host URL is `api.sourceclear.io`. If you use Veracode Platform API credentials, the host URL is `api.veracode.com/srcclr`.

XML APIs

Using the Veracode XML APIs

The Veracode Application Programming Interfaces (APIs) and their wrappers automate the actions involved in testing. You can use plugins to extend your workflow to seamlessly include Veracode security scanning.

Veracode provides an XML API for every task involved in scanning with Veracode. The following APIs and wrappers enable you to automate most of the tasks involved in scanning your applications.

All the API information is available as a PDF you can [download](#).

API Wrappers	Veracode provides API wrappers for Java and C#. Veracode recommends using API wrappers when working with the Veracode XML APIs.
Upload API	Use the Upload API to automatically send new builds of your applications to the Veracode Platform for static analysis scans.
Results API	Use the Results API to get a list of available applications and retrieve detailed application results.
Mitigation and Comments API	Integrate flaw comments and mitigation workflow tasks into IDEs and bug tracking systems.
Admin API	Use the Admin API to create and manage users and teams in the Veracode Platform.
Flaw Report API	The Flaw Report API creates a report that lists all fixed and unfixed flaws for the specified applications and/or scan type.
DynamicDS APIs	Several APIs are available to automate your DynamicDS scans.
VAST APIs	The Veracode VAST program has APIs for automating vendor and enterprise tasks.
Sandbox APIs	Use the Sandbox API calls to automate creating, updating, deleting, listing, and promoting developer sandboxes.

Upload APIs

Using the Upload API

You can use the Veracode Upload API to create an application, upload binary modules, check prescan results, and submit a static scan request.

For example, you can call the Upload API from your build system to automatically send a new build to Veracode for analysis as soon as the build finishes.

The Veracode Upload API is a basic HTTPS-based request API that uses simple HTTP calls. For performance reasons, the Upload API automatically compresses large XML files using Gzip, if your requesting tool supports it. Veracode recommends that you use a user agent that supports Gzip to access the Upload API. To learn about how to use the Upload API, read the [tutorial](#).

Prerequisites

Before using the Upload API, you must meet these prerequisites:

- A Veracode non-human [API user account](#) with the [Upload and Scan API role](#), or a Veracode human user account with one of the following roles:
 - If using the Visual Studio extensions, you need the [Upload and Scan API role](#) to add a new application.
 - The [Creator role](#) can create an application profile.
 - The [Submitter role](#) can submit a scan request.
 - The [Security Lead role](#) can perform all tasks.
- [Veracode API ID and key credentials](#)


Upload API Calls

The Upload API uses the following calls to automate tasks. Use these calls in your HTTP requests.

API Call	Description
beginprescan.do	Initiates the prescan of an application.
beginscan.do	Initiates the full scan of an application.
createapp.do	Creates a new application in the portfolio.
createbuild.do	Creates a build of an existing application.
deleteapp.do	Deletes an existing application from the portfolio.
deletebuild.do	Deletes the most recent build of an application.
getappinfo.do	Returns all the application profile information.
getapplist.do	Compiles a list of applications in the portfolio.
getbuildinfo.do	Returns all the information about the build.
getbuildlist.do	Compiles a list of all the builds of an application.
getfilelist.do	Compiles a list of the uploaded files in an application.
getpolicylist.do	Compiles a list of the policies available for use by your account.
getprescanresults.do	Fetches the results of the prescan.
getvendorlist.do	Compiles a list of third-party vendors provisioned for scans.
removefile.do	Removes specified files of an application.
updateapp.do	Amends an existing application in the portfolio.
updatebuild.do	Updates the most recent build of an application.
uploadfile.do	Uploads the files of an application.
uploadlargefile.do	Uploads the files of an application as a set of parts to avoid timeout errors during the upload.

Mapping Upload API Tasks

To understand how the Upload API works and in which order you use the API calls, the following table maps the API calls to the manual platform processes.

Step	Using the Veracode Platform	Using the Upload API	Returns XML File
1	Create an application profile.	createapp.do Optionally, if you want to name a scan, use createbuild.do.	appinfo.xml
2	Select and upload binaries.	uploadfile.do or uploadlargefile.do, beginprescan.do  NOTE: Use uploadfile.do or uploadlargefile.do in a programmatic loop to iterate through a directory in the local file system.	filelist.xml, buildinfo.xml
3	Wait for the prescan to complete.	getprescanresults.do or getbuildinfo.do	prescanresults.xml
4	Select and upload binary modules. Submit scan.	beginscan.do	buildinfo.xml

Click [here](#) to see all the calls the Upload API uses.

beginprescan.do

The `beginprescan.do` call runs the prescan of the application.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

The `beginprescan` call determines whether the auto-scan feature is on or off based on:

- The value of the `auto_scan` parameter.
- The `auto_scan` setting of the previous scan.
- If it is the first scan of an application, the `auto_scan` setting is based on the `auto_scan` feature switch value.

When `auto_scan` is true, the specified build automatically begins scanning after the prescan completes.

Resource URL

<https://analysiscenter.veracode.com/api/5.0/beginprescan.do>

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		
auto_scan	Boolean	<p>If you want to automatically submit a full scan, the auto_scan parameter is not present, auto_scan is set to true. If the auto_scan parameter is present, the full scan does not submit if the prescan is successful.</p> <ul style="list-style-type: none"> auto_scan setting of previous scans Feature switch value of auto_scan <p>The full scan does not submit if the prescan is successful.</p>
sandbox_id	Integer	Target sandbox ID.
scan_all_nonfatal_top_level_modules	Boolean	<p>If auto_scan is false, this parameter is ignored. If auto_scan is true, this parameter is required. The scan starts after prescan as long as the scan is successful, the module, and at least one of the top-level modules is successful.</p> <p>Default is false.</p>

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/beginprescan.do" "app_id=<app
id>" "auto_scan==false"
```

HTTPIe Results

The beginprescan.do call returns the buildinfo XML document, which references the [buildinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the buildinfo.xsd [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<buildinfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/4.0/buildinfo"

xsi:schemaLocation="https://analysiscenter.veracode.com
/schema/4.0/buildinfo

https://analysiscenter.veracode.com/resource/4.0/buildinfo.xsd"

buildinfo_version="<version #>" account_id="<account id>"
app_id="<app id>" build_id="<build id>"
<build version="v1" build_id="<build id>"
submitter="<VeracodeUsername>" platform="Not Specified"
```

```

        lifecycle_stage="Not Specified" results_ready="false"
policy_name="Veracode Transitional Very High"
        policy_version="1" policy_compliance_status="Not Assessed"
rules_status="Not Assessed"
        grace_period_expired="false" scan_overdue="false"
legacy_scan_engine="false">
    <analysis_unit analysis_type="Static" status="Pre-Scan
Submitted"/>
</build>
</buildinfo>

```

Java Example

```

java -jar vosp-api-wrappers-java-<version #>.jar -vid <VeracodeApiId>
-vkey <VeracodeApiKey> -action beginprescan -appid <app id>

```

Java Results

The `beginprescan.do` call returns the `buildinfo` XML document, which references the [buildinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `buildinfo.xsd` [schema documentation](#).

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<buildinfo xmlns="https://analysiscenter.veracode.com/schema/4.0/
buildinfo"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    account_id=<account id> app_id=<app id> build_id=<build id>
buildinfo_version=<version #>
    xsi:schemaLocation="https://analysiscenter.veracode.com/
schema/4.0/buildinfo
https://analysiscenter.veracode.com/resource/4.0/
buildinfo.xsd">
    <build build_id=<build id> grace_period_expired="false"
legacy_scan_engine="false" lifecycle_stage="Not Specified"
    platform="Not Specified" policy_compliance_status="Not
Assessed" policy_name="Veracode Recommended Very High"
    policy_version="1" results_ready="false" rules_status="Not
Assessed" scan_overdue="false"
    submitter=<VeracodeUsername> version="4 Dec 2018 Static">
    <analysis_unit analysis_type="Static" status="Pre-Scan
Submitted"/>
    </build>
</buildinfo>

```

beginscan.do

The `beginscan.do` call runs a full scan of the application. Specify the application ID and one of the four scan-type parameters.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/5.0/beginscan.do>

Parameters

Name	Type	Description
app_id Required	Integer	Application ID.
modules Required	String	Use either this parameter or one of: <ul style="list-style-type: none"> <code>scan_all_top_level_modules</code> <code>scan_selected_modules</code> <code>scan_previously_selected_modules</code> Comma-separated list of top-level module IDs for the specific scan in the results of <code>getprescan</code> . A module is associated with an <code>is_dependent</code> if it is a top-level module.
scan_all_top_level_modules Required	Boolean	Use either this parameter or one of: <ul style="list-style-type: none"> <code>modules</code> <code>scan_selected_modules</code> <code>scan_previously_selected_modules</code> Veracode recommends that you use the <code>scan_all_top_level_modules</code> parameter if you want to ensure the scan covers all errors, such as unsupported frameworks. <p>The top-level modules are the binaries that are directly loaded into the application. All the other binaries are loaded by these top-level modules. In Java, the uplevel modules are almost always the top-level modules. In .NET, the DLLs are almost always the top-level modules. In other languages, the top-level modules are the binaries that are directly loaded into the application.</p>
scan_selected_modules Required	Boolean	Use either this parameter or one of: <ul style="list-style-type: none"> <code>modules</code> <code>scan_all_top_level_modules</code> <code>scan_previously_selected_modules</code> When this parameter is true, only the modules selected in the Platform UI are scanned. This selection may be different from <code>scan_all_top_level_modules</code> , depending on the party modules are selected and any top-level modules are selected.
scan_previously_selected_modules Required	Boolean	Use either this parameter or one of: <ul style="list-style-type: none"> <code>modules</code> <code>scan_all_top_level_modules</code> <code>scan_selected_modules</code> When true, only the modules selected in the previous scan outcome may or may not be the same as using <code>scan_all_top_level_modules</code> , depending on the party modules are selected and any top-level modules are selected.
sandbox_id	Integer	Target sandbox ID.

HTTPIe Examples

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/beginscan.do" "app_id==<app id>"
"scan_all_top_level_modules==true"

http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/beginscan.do" "app_id==<app id>"
"modules==<module id>,<module id>"
```

HTTPIe Results

The beginscan.do call responds with the buildinfo XML document, which references the [buildinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the buildinfo.xsd [schema documentation](#).

Response for the scan_all_top_level_modules example:

```
<?xml version="1.0" encoding="UTF-8"?>

<buildinfo xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;4.0&#x2f;buildinfo"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;4.0&#x2f;
  buildinfo
https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
4.0&#x2f;buildinfo.xsd"
  buildinfo_version="1.4" account_id="<account id>" app_id="<app
id>" build_id="<build id>"
  <build version="<build name>"
    build_id="<build id>" submitter="<VeracodeUsername>"
platform="Not Specified" lifecycle_stage="Not Specified"
  results_ready="false" policy_name="Veracode Recommended Very
High" policy_version="1"
  policy_compliance_status="Not Assessed" rules_status="Not
Assessed" grace_period_expired="false"
  scan_overdue="false" legacy_scan_engine="false"
launch_date="2019-08-22T14&#x3a;27&#x3a;59-04&#x3a;00">
  <analysis_unit analysis_type="Static" status="Submitted to
Engine" engine_version="20190805180615"/>
  </build>
</buildinfo>
```

Response for the modules example:

```
<?xml version="1.0" encoding="UTF-8"?>

<buildinfo xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"
```

```

xmlns="https://analysiscenter.veracode.com/schema/4.0/buildinfo"

xsi:schemaLocation="https://analysiscenter.veracode.com/schema/4.0/buildinfo
https://analysiscenter.veracode.com/resource/4.0/buildinfo.xsd"
  buildinfo_version="1.4" account_id="<account id>" app_id="<app id>" build_id="<build id>"
  <build version="22 Aug 2019 Static" build_id="build id"
  submitter="<VeracodeUsername>" platform="Not Specified"
  lifecycle_stage="Not Specified" results_ready="false"
  policy_name="Veracode Recommended Very High"
  policy_version="1" policy_compliance_status="Conditional Pass"
  policy_updated_date="2019-08-22T14:42:38-04:00"
  rules_status="Pass" grace_period_expired="false"
  scan_overdue="false" legacy_scan_engine="false"
  launch_date="2019-08-22T14:27:59-04:00">
    <analysis_unit analysis_type="Static" status="Submitted to
    Engine" engine_version="20190805180615"/>
  </build>
</buildinfo>

```

If no selected modules exist in the Veracode Platform and the call uses the `scan_selected_modules` parameter, the return contains:

```

<?xml version="1.0" encoding="UTF-8"?>
  <error>No modules parameter specified</error>

```

Java Example

```

java -jar vosp-api-wrappers-java-<version #>.jar -vid <VeracodeApiId>
-vkey <VeracodeApiKey> -action beginscan -appid <app id> -toplevel
true

```

Java Results

The `beginscan.do` call responds with the `buildinfo` XML document, which references the [buildinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the [buildinfo.xsd schema documentation](#).

```

<?xml version="1.0" encoding="UTF-8"?>

<buildinfo xmlns="https://analysiscenter.veracode.com/schema/4.0/
buildinfo"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  account_id="<account id>" app_id="<app id>" build_id="<build id>"
  buildinfo_version="1.4"
  xsi:schemaLocation="https://analysiscenter.veracode.com/
schema/4.0/buildinfo
https://analysiscenter.veracode.com/resource/4.0/
buildinfo.xsd">
  <build build_id="<build id>" grace_period_expired="false"
  legacy_scan_engine="false"
  lifecycle_stage="Not Specified" platform="Not Specified"

```

```

policy_compliance_status="Not Assessed"
  policy_name="Veracode Recommended Very High" policy_version="1"
results_ready="false"
  rules_status="Not Assessed" scan_overdue="false"
submitter="JoeUser" version="4 Dec 2018 Static">
  <analysis_unit analysis_type="Static" engine_version="131771"
status="Submitted to Engine"/>
</build>
</buildinfo>

```

createapp.do

The `createapp.do` call creates a new application in the portfolio.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/5.0/createapp.do>

Parameters

Name	Type	Description
app_name Required	String	Your application name.
business_criticality Required	String (case-sensitive)	Matches the names of Business Criticality <ul style="list-style-type: none"> • Very High • High • Medium • Low • Very Low
description	String	Description of the target application.
vendor_id	Integer	Indicates a third-party (COTS) application. It validates against the vendor ID set as default for the business criticality policy.
policy	String (case-sensitive)	Validates against the names of existing policies for the business criticality policy set as default for the business criticality policy.
business_unit	String (case-sensitive)	Validates against the names of existing business units.
business_owner	String	Name of the business owner for the application.
business_owner_email	String	Email for the business owner of the application.
teams	String	Comma-separated list of team names. Validates against the teams for this account.
origin	String (case-sensitive)	Validates against the names of the Origin enumeration <ul style="list-style-type: none"> • 3rd party library • Purchased Application • Contractor

Name	Type	Description
		<ul style="list-style-type: none"> Internally Developed Open Source Outsourced Team
industry	String (case-sensitive)	Validates against the names of the Industry Parameter Values . Defaults to the same industry as the account.
app_type	String (case-sensitive)	Validates against the names of the Application Parameter Values .
deployment_method	String (case-sensitive)	Validates against the names of the Deployment Parameter Values . <ul style="list-style-type: none"> Web Based Enterprise Application Enhancement Client/Server Mobile Stand Alone
web_application	Boolean	Default is false.
archer_app_name	String	Name of the application in Archer.
tags	String	Comma-separated list of tags.
next_day_scheduling_enabled	Boolean	Specifies if a user can schedule next-day consultations. Veracode human user accounts with the Security role and to non-human API accounts with the Upstream role . Default is false.

Industry Parameter Values

Values are case-sensitive.

- Aerospace
- Agriculture
- Apparel
- Automotive and Transport
- Banking
- Beverages
- Biotechnology
- Business Services
- Charitable Organizations
- Chemicals
- Communications
- Computer Hardware
- Consulting
- Construction
- Consumer Products Manufacturers
- Consumer Services
- Cultural Institutions
- Food & Beverage
- Foundations
- Government
- Healthcare
- Hospitality
- Insurance
- Manufacturing
- Machinery
- Media & Entertainment
- Membership Organizations
- Metals and Mining
- Other
- Pharmaceuticals
- Real Estate
- Recreation
- Retail
- Security Products and Services

- Education
- Electronics
- Energy
- Engineering
- Environmental
- Finance
- Software
- Technology
- Telecommunications Equipment
- Telecommunications
- Transportation
- Utilities

app_type Parameter Values

Values are case-sensitive.

- Application Design/Construction/IDE/Analysis
- Application Life-Cycle Management
- Application Server/Integration Server
- Back-Office Enterprise
- CRM/Groupware/Messaging
- Consumer
- Content Management/Authoring
- Engineering
- Enterprise Resource Planning
- Information Access/Delivery/Mining/Portal
- Information/Data Management/Database
- Middleware/Message-oriented/Transaction
- Network Management
- Networking
- Other
- Other Development Tools
- Security
- ServerWare/Clustering/Web/VM
- Storage
- System-Level Software
- Systems Management
- Testing Tools

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/createapp.do" "app_name==<app
name>" "business_criticality==Very High"
```

HTTPIe Results

The createapp.do call returns the appinfo XML document, which references the [appinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the [appinfo.xsd schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<appinfo xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
```

```
#x2f;2.0&#x2f;appinfo"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;2.0&#x2f;appinfo

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
2.0&#x2f;appinfo.xsd" appinfo_version="1.1"
  account_id="<account id>"
  <application app_id="<app id>" app_name="<app name>"
business_criticality="Very High"
  policy="Veracode Recommended Very High" teams="" origin="Not
Specified" industry_vertical="Not Specified"
  app_type="Not Specified" deployment_method="Not Specified"
is_web_application="false"
  modified_date="2019-08-15T13&#x3a;56&#x3a;46-04&#x3a;00"
cots="false" vast="false"
  business_unit="Not Specified" tags=""
  <customfield name="Custom 1" value="100"/>
  <customfield name="Custom 2" value="Deferred"/>
  <customfield name="Custom 3" value=""/>
  <customfield name="Custom 4" value=""/>
  <customfield name="Custom 5" value=""/>
  <customfield name="Custom 6" value=""/>
  <customfield name="Custom 7" value=""/>
  <customfield name="Custom 8" value=""/>
  <customfield name="Custom 9" value=""/>
  <customfield name="Custom 10" value=""/>
  </application>
</appinfo>
```

Java Example

```
java -jar vosp-api-wrappers-java-<version #>.jar -vid <VeracodeApiId>
-vkey <VeracodeApiKey> -action createapp -appname MyApp -criticality
veryhigh
```

Java Results

The createapp.do call returns the appinfo XML document, which references the [appinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the [appinfo.xsd schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<appinfo xmlns="https://analysiscenter.veracode.com/schema/2.0/
appinfo"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  account_id="<account id>" appinfo_version="1.1"
  xsi:schemaLocation="https://analysiscenter.veracode.com/
schema/2.0/appinfo https://analysiscenter.veracode.com/resource/2.0/
appinfo.xsd">
  <application app_id="<app id>" app_name="MyApp" app_type="Not
Specified" business_criticality="Very High" business_unit="Not
Specified"
    cots="false" deployment_method="Not Specified"
    industry_vertical="Not Specified" is_web_application="false"
    modified_date="2018-12-04T11:18:38-05:00" origin="Not
```

```

Specified" policy="Veracode Recommended Very High" tags="" teams=""
vast="false">
  <customfield name="Custom 1" value="100"/>
  <customfield name="Custom 2" value="Deferred"/>
  <customfield name="Custom 3" value=""/>
  <customfield name="Custom 4" value=""/>
  <customfield name="Custom 5" value=""/>
  <customfield name="Custom 6" value=""/>
  <customfield name="Custom 7" value=""/>
  <customfield name="Custom 8" value=""/>
  <customfield name="Custom 9" value=""/>
  <customfield name="Custom 10" value=""/>
</application>
</appinfo>

```

createbuild.do

The `createbuild.do` call creates a new build of an existing application in the portfolio.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/5.0/createbuild.do>

Parameters

The [uploadfile.do](#) call also creates a build. Therefore, it is not necessary to use this call as part of the [Upload API](#) workflow.

Name	Type	Description
app_id	Integer	Application ID.
Required		
version	String	Specify a unique identifier for the build.
lifecycle_stage	String (case-sensitive)	Validates against the names of the Lifecycle enums: <ul style="list-style-type: none"> In Development. This category includes pre-Alpha. Internal or Alpha Testing External or Beta Testing Deployed. This category includes in production and ac Maintenance. This category includes only bug fixes. Cannot Disclose Only used if lifecycle_stage_id is not provided.
launch_date	String	Validates against the mm/dd/yyyy date format.
sandbox_id	Integer	Target sandbox ID.
platform	String	This parameter is not supported but maintained for backwa
Deprecated		against the Platform enums. Only used if platform_id

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/createbuild.do" "app_id==<app
id>" "version==<version name>"
```

HTTPIe Results

The createbuild.do call returns the buildinfo XML document, which references the [buildinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the buildinfo.xsd [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<buildinfo xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;4.0&#x2f;buildinfo"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;4.0&#x2f;buildinfo

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
4.0&#x2f;buildinfo.xsd" buildinfo_version="1.4"
  account_id="<account id>" app_id="<app id>"
  sandbox_id="<sandbox id>" build_id="<build id>"><build
version="<build name>"
  build_id="<build id>" submitter="<VeracodeUsername>"
platform="Not Specified" lifecycle_stage="Not Specified"
  results_ready="false" policy_name="Veracode Transitional Very
High" policy_version="1" policy_compliance_status="Not Assessed"
  rules_status="Not Assessed" grace_period_expired="false"
scan_overdue="false" legacy_scan_engine="false">
  <analysis_unit analysis_type="Static" status="Incomplete"/>
</build>
</buildinfo>
```

deleteapp.do

The deleteapp.do call deletes an existing application in the portfolio.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/5.0/deleteapp.do>

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/deleteapp.do" "app_id=<app id>"
```

HTTPIe Results

The deleteapp.do call returns the applist XML document, which references the [applist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the [applist.xsd schema documentation](#).

The return lists the remaining applications after the successful deletion.

```
<?xml version="1.0" encoding="UTF-8"?>

<applist xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;2.0&#x2f;applist"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;2.0&#x2f;applist

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
2.0&#x2f;applist.xsd" applist_version="1.2"
  account_id="<account id>"><app app_id="<app id>" app_name="<app
name>"
  policy_updated_date="2019-08-13T14&#x3a;09&#x3a;11-04&#x3a;00"/>
  <app app_id="app id" app_name="<app name>"
policy_updated_date="2019-08-13T14&#x3a;03&#x3a;33-04&#x3a;00"/>
  <app app_id="app id" app_name="<app name>"
policy_updated_date="2019-08-13T14&#x3a;03&#x3a;33-04&#x3a;00"/>
  <app app_id="app id" app_name="<app name>"
policy_updated_date="2019-08-13T14&#x3a;03&#x3a;33-04&#x3a;00"/>
</applist>
```

deletebuild.do

The deletebuild.do call deletes the most recent static build of an existing application in the portfolio.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/deletebuild.do`

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		
sandbox_id	Integer	The ID of the sandbox that contains the static build to delete.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/deletebuild.do" "app_id=<app id>"
```

HTTPIe Results

The deletebuild.do call returns the buildlist XML document, which references the [buildlist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the buildlist.xsd [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<buildlist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/2.0/buildlist"

xsi:schemaLocation="https://analysiscenter.veracode.com
/schema/2.0/buildlist

https://analysiscenter.veracode.com/resource/2.0/buildlist.xsd" buildlist_version="1.3"
  account_id="<account id>" app_id="<app id>" app_name="<app
name>">
</buildlist>
```

getappinfo.do

The getappinfo.do call provides information about the application.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/getappinfo.do`

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getappinfo.do" "app_id=<app id>"
```

HTTPIe Results

The getappinfo.do call returns the appinfo XML document, which references the [appinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the appinfo.xsd [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<appinfo xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;2.0&#x2f;appinfo"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;2.0&#x2f;appinfo

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
2.0&#x2f;appinfo.xsd" appinfo_version="1.1"
  account_id="<account id>"
    <application app_id="<app id>" app_name="<app name>"
description="<app description>" business_criticality="Very High"
  policy="Veracode Transitional Very High"
policy_updated_date="2019-08-13T14&#x3a;02&#x3a;08-04&#x3a;00"
  teams="Demo Team" origin="Not Specified"
industry_vertical="Other" app_type="Other" deployment_method="Not
Specified"
  is_web_application="false" archer_app_name="<archer app name>"
modified_date="2019-08-15T11&#x3a;27&#x3a;47-04&#x3a;00"
  cots="false" vast="false" business_unit="Not Specified" tags="">
  <customfield name="Custom 1" value="" />
  <customfield name="Custom 2" value="" />
  <customfield name="Custom 3" value="" />
  <customfield name="Custom 4" value="" />
  <customfield name="Custom 5" value="" />
  <customfield name="Custom 6" value="" />
  <customfield name="Custom 7" value="" />
  <customfield name="Custom 8" value="" />
  <customfield name="Custom 9" value="" />
  <customfield name="Custom 10" value="foo" />
```

```
</application>
</appinfo>
```

Java Example

```
java -jar vosp-api-wrappers-java-<version #>.jar -vid <VeracodeApiId>
-vkey <VeracodeApiKey> -action getappinfo -appid <app id>
```

Java Results

The `getappinfo.do` call returns the `appinfo` XML document, which references the [appinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `appinfo.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<appinfo xmlns="https://analysiscenter.veracode.com/schema/2.0/"
appinfo"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  account_id="<account info>" appinfo_version="1.1"
  xsi:schemaLocation="https://analysiscenter.veracode.com/
schema/2.0/appinfo
  https://analysiscenter.veracode.com/resource/2.0/appinfo.xsd">
  <application app_id="<app id>" app_name="<app name>"
app_type="Other" archer_app_name="2501"
  business_criticality="High"
  business_unit="Not Specified" cots="false"
deployment_method="Not Specified"
  description="MyApp is a teaching web applications."
industry_vertical="Other"
  is_web_application="false"
modified_date="2018-06-18T10:25:40-04:00" origin="Open Source"
  policy="Scan Policy"
policy_updated_date="2018-11-04T23:29:42-05:00" tags="" teams=""
vast="false">
  <customfield name="Custom 1" value=""/>
  <customfield name="Custom 2" value=""/>
  <customfield name="Custom 3" value=""/>
  <customfield name="Custom 4" value=""/>
  <customfield name="Custom 5" value=""/>
  <customfield name="Custom 6" value=""/>
  <customfield name="Custom 7" value=""/>
  <customfield name="Custom 8" value=""/>
  <customfield name="Custom 9" value=""/>
  <customfield name="Custom 10" value=""/>
  </application>
</appinfo>
```

getapplist.do

The `getapplist.do` call compiles a list of the applications in the portfolio. If you use the optional parameter to include user information, this call also returns details about tasks you have permissions to perform, such as view scan results and approve mitigations.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/getapplist.do`

Parameters

Name	Type	Description
<code>include_user_info</code>	Boolean	If true, the list of applications concludes with information about the user and the assigned permissions. Default is false.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getapplist.do"
"include_user_info==true"
```

HTTPIe Results

The `getapplist.do` call returns the `applist` XML document, which references the [applist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the [applist.xsd schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<applist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/2.0/applist"

xsi:schemaLocation="https://analysiscenter.veracode.com/schema/2.0/applist

https://analysiscenter.veracode.com/resource/2.0/applist.xsd" applist_version="1.2"
  account_id="<account id>"
    <app app_id="<app id>" app_name="<app name>"
policy_updated_date="2019-08-13T14:09:11-04:00"/>
    <app app_id="<app id>" app_name="<app name>"
policy_updated_date="2019-07-13T14:03:33-04:00"/>
    <app app_id="<app id>" app_name="<app name>"
policy_updated_date="2019-08-16T14:03:33-11:00:00"/>
    <app app_id="<app id>" app_name="<app name>"
policy_updated_date="2019-09-03T14:03:33-07:00:00"/>
    <user login_account_type="user" username="<VeracodeUsername>"
create_application_profile="true" create_sandbox="true"
  create_new_build="true" create_policy_scan="true"
create_sandbox_scan="true" assign_app_to_team="true"
  assign_app_to_any_team="true" view_sandbox="true"
view_results="true" approve_mitigations="true"
  submit_static_scan="true" submit_policy_static_scan="true"
```

```
submit_sandbox_static_scan="true"/>
</applist>
```

Java Example

```
java -jar vosp-api-wrappers-java-<version #>.jar -vid <VeracodeApiId>
-vkey <VeracodeApiKey> -action getapplist
```

Java Results

The `getapplist.do` call returns the `applist` XML document, which references the [applist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the [applist.xsd schema documentation](#).

```
<applist xmlns="https://analysiscenter.veracode.com/schema/2.0/
applist"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  account_id="12345"
  applist_version="1.2" xsi:schemaLocation="https://
analysiscenter.veracode.com/schema/2.0/applist
https://analysiscenter.veracode.com/resource/2.0/applist.xsd">
  <app app_id="<app id>" app_name="<app name>"
  policy_updated_date="2017-11-16T13:55:05-05:00"/>
  <app app_id="<app id>" app_name="<app name>"
  policy_updated_date="2018-08-17T02:24:25-04:00"/>
  <app app_id="<app id>" app_name="<app name>" />
</applist>
```

getbuildinfo.do

The `getbuildinfo.do` call provides information about the most recent scan or a specific scan of the application.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/5.0/getbuildinfo.do>

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		
build_id	Integer	Application or sandbox build ID. Default is the most recent s
sandbox_id	Integer	Target sandbox ID.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getbuildinfo.do" "app_id=<app
id>"
```

HTTPIe Results

The `getbuildinfo.do` call returns the `buildinfo` XML document, which references the [buildinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `buildinfo.xsd` [schema documentation](#).

For information on the build status messages, see [API Build Status Information](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<buildinfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/4.0/buildinfo"

xsi:schemaLocation="https://analysiscenter.veracode.com
/schema/4.0/buildinfo

https://analysiscenter.veracode.com/resource/4.0/buildinfo.xsd" buildinfo_version="1.4"
  account_id="<account id>" app_id="<app id>" build_id="<build
id>">
  <build version="13 Aug 2019 Static" build_id="<build id>"
submitter="Veracode" platform="Not Specified"
  lifecycle_stage="Not Specified" results_ready="false"
policy_name="Veracode Transitional Very High" policy_version="1"
  policy_compliance_status="Not Assessed"
policy_updated_date="2019-08-13T14:02:04+00:00"
  rules_status="Not Assessed" grace_period_expired="false"
scan_overdue="false" legacy_scan_engine="false">
    <analysis_unit analysis_type="Static" status="Scan In Process"
engine_version="20190805180615"/>
  </build>
</buildinfo>
```

Java Example

```
java -jar vosp-api-wrappers-java-<version #>.jar -vid <VeracodeApiId>
-vkey <VeracodeApiKey> -action getbuildinfo -appid <app id>
```

Java Results

The `getbuildinfo.do` call returns the `buildinfo` XML document, which references the [buildinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the

buildinfo.xsd [schema documentation](#). For information on the build status messages, see [API Build Status Information](#).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<buildinfo xmlns="https://analysiscenter.veracode.com/schema/4.0/buildinfo"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  account_id="<account id>" app_id="<app id>" build_id="<build id>" buildinfo_version="1.4"
  xsi:schemaLocation="https://analysiscenter.veracode.com/schema/4.0/buildinfo
  https://analysiscenter.veracode.com/resource/4.0/buildinfo.xsd">
  <build build_id="<build id>" grace_period_expired="false"
  legacy_scan_engine="false"
  lifecycle_stage="Not Specified" platform="Not Specified"
  policy_compliance_status="Did Not Pass"
  policy_name="Veracode Transitional Very High"
  policy_updated_date="2019-08-13T14:09:11-04:00"
  policy_version="1" results_ready="true" rules_status="Did Not Pass" scan_overdue="false"
  submitter="Veracode" version="13 Aug 2019 Static">
    <analysis_unit analysis_type="Static"
    engine_version="20190805180615"
    published_date="2019-08-13T14:08:35-04:00"
    published_date_sec="1565719715" status="Results Ready"/>
  </build>
</buildinfo>
```

getbuildlist.do

The `getbuildlist.do` call produces a list of the application policy or sandbox scans in progress or already complete.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/getbuildlist.do`

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		
sandbox_id	Integer	Target sandbox ID.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getbuildlist.do" "app_id=<app
id>"
```

HTTPIe Results

The `getbuildlist.do` call returns the `buildlist` XML document, which references the [buildlist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `buildlist.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<buildlist xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;2.0&#x2f;buildlist"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;2.0&#x2f;buildlist

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
2.0&#x2f;buildlist.xsd" buildlist_version="1.3"
  account_id="<account id>" app_id="<app id>" app_name="<app
name>"><build build_id="<build id>"
  version="13 Aug 2019 Static"
  policy_updated_date="2019-08-13T14&#x3a;02&#x3a;08-04&#x3a;00"/>
</buildlist>
```

Java Example

```
java -jar vosp-api-wrappers-java-<version #>.jar -vid <VeracodeApiId>
-vkey <VeracodeApiKey> -action getbuildlist -appid <app id>
```

Java Results

The `getbuildlist.do` call returns the `buildlist` XML document, which references the [buildlist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `buildlist.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<buildlist xmlns="https://analysiscenter.veracode.com/schema/2.0/
buildlist"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  account_id="74370"
  app_id="<app id>" app_name="<app name>" buildlist_version="1.3"
  xsi:schemaLocation="https://analysiscenter.veracode.com/
schema/2.0/buildlist
```

```

https://analysiscenter.veracode.com/resource/2.0/buildlist.xsd">
<build build_id="<build id>"
  policy_updated_date="2019-08-13T14:09:11-04:00" version="13 Aug
2019 Static"/>
</buildlist>

```

getfilelist.do

The `getfilelist.do` call compiles a list of files uploaded for a static scan. The returned XML provides MD5 checksums for these files if available.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/getfilelist.do`

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		
build_id	Integer	Application or sandbox build ID. Default is the most recent s
sandbox_id	Integer	Create a list of files from the sandbox with this ID.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```

http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getfilelist.do" "app_id==<app
id>" "build_id==<build id>"

```

HTTPIe Results

The `getfilelist.do` call returns the `filelist` XML document, which references the [filelist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `filelist.xsd` [schema documentation](#).

```

<?xml version="1.0" encoding="UTF-8"?>

<filelist xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;2.0&#x2f;filelist"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;2.0&#x2f;filelist

```

```
https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
2.0&#x2f;filelist.xsd" filelist_version="1.1"
  account_id="<account id>" app_id="<app id>" build_id="<build
id>">
  <file file_id="<file id>" file_name="<file name>"
file_status="Uploaded" file_md5="<file md5>"/>
  <file file_id="<file id>" file_name="<file name>"
file_status="Uploaded" file_md5="<file md5>"/>
</filelist>
```

Java Example

```
java -jar vosp-api-wrappers-java-<version #>.jar -vid <VeracodeApiId>
-vkey <VeracodeApiKey> -action getfilelist -appid <app id>
```

Java Results

The `getfilelist.do` call returns the `filelist` XML document, which references the [filelist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `filelist.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<filelist xmlns="https://analysiscenter.veracode.com/schema/2.0/
filelist"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  account_id="<account id>" app_id="<app id>" build_id="<build
id>" filelist_version="1.1"
  xsi:schemaLocation="https://analysiscenter.veracode.com/
schema/2.0/filelist
https://analysiscenter.veracode.com/resource/2.0/filelist.xsd">
  <file file_id="<file id>" file_md5="<file md5>"
    file_name="<file name>" file_status="Uploaded"/>
</filelist>
```

getpolicylist.do

The `getpolicylist.do` call compiles a list of policies available to your account.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/getpolicylist.do`

Parameters

This call takes no parameters.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getpolicylist.do"
```

HTTPIe Results

The `getpolicylist.do` call returns the `list` XML document, which references the [policylist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `policylist.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<policylist xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;4.0&#x2f;policylist"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;4.0&#x2f;policylist

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
4.0&#x2f;policylist.xsd"
  account_id="<account id>">
    <policies names="Veracode Recommended Very High,Veracode
Recommended High,Veracode Recommended Medium,
Veracode Recommended Low,Veracode Recommended Very Low,Veracode
Recommended Mobile Policy,
Veracode Recommended Very High &#x2b; SCA,Veracode Recommended
High &#x2b; SCA,Veracode Recommended Medium &#x2b;
SCA,Veracode Transitional Very High,Veracode Transitional
High,Veracode Transitional Medium,Veracode Transitional Low,
Veracode Transitional Very Low,PCI 3.2.1"/>
  </policies>
</policylist>
```

Java Example

```
java -jar vosp-api-wrappers-java-<version #>.jar -vid <VeracodeApiId>
-vkey <VeracodeApiKey> -action getpolicylist
```

Java Results

The `getpolicylist.do` call returns the `list` XML document, which references the [policylist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `policylist.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<policylist xmlns="https://analysiscenter.veracode.com/schema/4.0/
policylist"
```

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
account_id="<account_id>
xsi:schemaLocation="https://analysiscenter.veracode.com/
schema/4.0/policylist
https://analysiscenter.veracode.com/resource/4.0/
policylist.xsd">
  <policies names="Veracode Recommended Very High,Veracode
Recommended High,Veracode Recommended Medium,
Veracode Recommended Low, Veracode Recommended Very
Low,Veracode Recommended Mobile Policy,
Veracode Recommended Very High + SCA,Veracode Recommended High
+ SCA,Veracode Recommended Medium + SCA,
Veracode Transitional Very High,Veracode Transitional
High,Veracode Transitional Medium,
Veracode Transitional Low,Veracode Transitional Very Low,PCI
3.2"/>
</policylist>

```

getprescanresults.do

The `getprescanresults.do` call fetches the results of the prescan.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/getprescanresults.do`

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		
build_id	Integer	Application or sandbox build ID.
sandbox_id	Integer	The ID of the source sandbox for prescan results.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```

http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getprescanresults.do"
"app_id==<app id>"

```

HTTPIe Results

The `getprescanresults.do` call returns the `prescanresults` XML document, which references the [prescanresults.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `prescanresults.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<prescanresults xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;2.0&#x2f;prescanresults"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;2.0&#x2f;prescanresults

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
2.0&#x2f;prescanresults.xsd" prescanresults_version="1.4"
  account_id="<account id>" app_id="<app id>" build_id="<build
id>"><module id="<module id>" name="<app name>"
  app_file_id="<app file id>" checksum="<checksum>" platform="JVM
&#x2f; Java J2SE 8 &#x2f; JAVAC_8" size="0KB" status="OK"
  has_fatal_errors="false" is_dependency="false"> <issue
details="No supporting files or PDB files"/>
  </module>
  <module id="<module id>" name="JS files within <app id>"
app_file_id="<app file id>" platform="JAVASCRIPT &#x2f; JavaScript
&#x2f;
  JAVASCRIPT_5_1" size="8KB" status="OK" has_fatal_errors="false"
is_dependency="false">
    <issue details="No supporting files or PDB files"/>
  </module>
</prescanresults>
```

See [API Prescan Status Information](#) for more information.

Java Example

```
java -jar vosp-api-wrappers-java-<version #>.jar -vid <VeracodeApiId>
-vkey <VeracodeApiKey> -action getprescanresults -appid <app id>
```

Java Results

The `getprescanresults.do` call returns the `prescanresults` XML document, which references the [prescanresults.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `prescanresults.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<prescanresults xmlns="https://analysiscenter.veracode.com/schema/2.0/
prescanresults"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  account_id="<account id>" app_id="<app id>" build_id="<build
id>" prescanresults_version="1.4"
  xsi:schemaLocation="https://analysiscenter.veracode.com/
```

```

schema/2.0/prescanresults
  https://analysiscenter.veracode.com/resource/2.0/
prescanresults.xsd">
  <module app_file_id="<app file id>" checksum="<checksum>"
has_fatal_errors="false" id="1035970068"
    is_dependency="false" name="httpd" platform="IA32 / Red Hat
Enterprise Linux v4 (IA32) / GCC_Linux_IA32_3_4_6"
    size="15MB" status="OK">
    <file_issue details="Found (Optional)" filename="<filename>"/>
    <file_issue details="Found (Optional)" filename="<filename>"/>
  </module>
  <module app_file_id="<app file id>" checksum="<checksum>"
has_fatal_errors="false" id="1035970069"
    is_dependency="true" name="<filename>" platform="IA32 / Red Hat
Enterprise Linux v4 (IA32) / GCC_Linux_IA32_3_4_6"
    size="5MB" status="OK">
    <issue details="No supporting files or PDB files"/>
  </module>
  <module app_file_id="<app file id>" checksum="<checksum>"
has_fatal_errors="false" id="1035970070"
    is_dependency="true" name="filename" platform="IA32 / Red Hat
Enterprise Linux v4 (IA32) / GCC_Linux_IA32_3_4_6"
    size="8MB" status="OK">
    <file_issue details="Found (Optional)" filename="<filename>"/>
  </module>
</prescanresults>

```

getvendorlist.do

The `getvendorlist.do` call compiles a list of your organization vendors for third-party scans.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/getvendorlist.do`

Parameters

This call takes no parameters.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```

http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getvendorlist.do"

```


HTTPie Results

The `getvendorlist.do` call returns the `vendorlist` XML document, which references the [vendorlist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `vendorlist.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<vendorlist xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;2.0&#x2f;vendorlist"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;2.0&#x2f;vendorlist

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
2.0&#x2f;vendorlist.xsd"
  account_id="<account id>"
    <vendor_id="<vendor id>" vendor_name="<vendor list>"/>
    <vendor_id="<vendor id>" vendor_name="<vendor list>"/>
    <vendor_id="<vendor id>" vendor_name="<vendor list>"/>
  </vendorlist>
```

removefile.do

The `removefile.do` call deletes a file from an existing application build.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/removefile.do`

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		
file_id	Integer	Obtain the <code>file_id</code> by calling getfilelist.do on page 91.
Required		
sandbox_id	Integer	Enter the ID of the sandbox that contains the file to delete.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/removefile.do" "app_id=<app id>"
"file_id=<file id>"
```

HTTPIe Results

The `removefile.do` call returns the `filelist` XML document, which references the [filelist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the [filelist.xsd schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<filelist xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;2.0&#x2f;filelist"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;2.0&#x2f;filelist

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
2.0&#x2f;filelist.xsd" filelist_version="1.1"
    account_id="<account id>" app_id="<app id>"
    sandbox_id="<sandbox id>" build_id="<build id>"
</filelist>
```

updateapp.do

The `updateapp.do` call modifies the settings of an existing application in the portfolio.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/5.0/updateapp.do>

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		
app_name	String	Changes the application name.
description	String	Changes the application description.
business_criticality	String (case-sensitive)	Matches the names of the business criticality enums:

Name	Type	Description
		<ul style="list-style-type: none"> • Very High • High • Medium • Low • Very Low <p>You cannot change this value for applications that a v</p>
policy	String (case-sensitive)	<p>Validates against the names of existing policies for th</p> <p>value for applications that a vendor shares with your a</p> <p>Default is the policy set as default for the business_</p>
business_unit	String (case-sensitive)	Validates against the names of existing business unit
business_owner	String	Name of the business owner for the application.
business_owner_email	String	Email address for the business owner of the applicati
teams	String (case-sensitive)	Comma-separated list of team names. Validates again
origin	String (case-sensitive)	<p>Validates against the names of the Origin enums:</p> <ul style="list-style-type: none"> • 3rd party library • Purchased Application • Contractor • Internally Developed • Open Source • Outsourced Team <p>Default is Not Specified.</p>
industry	String (case-sensitive)	<p>Validates against the names of the Industry enums</p> <p>Defaults to the same industry as the account.</p>
app_type	String (case-sensitive)	Validates against the names of the Application P Parameter Values . Defaults to Not Specified.
deployment_method	String (case-sensitive)	<p>Validates against the names of the Deployment Me</p> <p>Specified. Case-sensitive enum values include:</p> <ul style="list-style-type: none"> • Web Based • Enterprise Application Enhancement • Client/Server • Mobile • Stand Alone
archer_app_name	String	Name of the application in Archer.
tags	String	Comma-separated list of tags.
custom_field_name	String (case-sensitive)	Specifies the custom field that custom_field_valu other is required. Call <code>updateapp.do</code> once for each

Name	Type	Description
custom_field_value	String	The value of the custom field that custom_field_name the other is required. Call updateapp.do once for each
next_day_scheduling_enabled	Boolean	Specifies if a user can schedule next-day consultation user accounts with the Security Lead or Administrator with the Upload and Scan API role. Default is false.

industry Parameter Values

Values are case-sensitive.

- Aerospace
- Agriculture
- Apparel
- Automotive and Transport
- Banking
- Beverages
- Biotechnology
- Business Services
- Charitable Organizations
- Chemicals
- Communications
- Computer Hardware
- Consulting
- Construction
- Consumer Products Manufacturers
- Consumer Services
- Cultural Institutions
- Education
- Electronics
- Energy
- Engineering
- Environmental
- Finance
- Food & Beverage
- Foundations
- Government
- Healthcare
- Hospitality
- Insurance
- Manufacturing
- Machinery
- Media & Entertainment
- Membership Organizations
- Metals and Mining
- Other
- Pharmaceuticals
- Real Estate
- Recreation
- Retail
- Security Products and Services
- Software
- Technology
- Telecommunications Equipment
- Telecommunications
- Transportation
- Utilities

app_type Parameter Values

Values are case-sensitive.

- Application Design/Construction/IDE/Analysis
- Application Life-Cycle Management
- Application Server/Integration Server
- Back-Office Enterprise
- CRM/Groupware/Messaging
- Middleware/Message-oriented/Transaction
- Network Management
- Networking
- Other
- Other Development Tools
- Security

- Consumer
- Content Management/Authoring
- Engineering
- Enterprise Resource Planning
- Information Access/Delivery/
Mining/Portal
- Information/Data Management/
Database
- ServerWare/Clustering/Web/VM
- Storage
- System-Level Software
- Systems Management
- Testing Tools

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/updateapp.do" "app_id==<app id>"
"custom_field_name==Custom 10" "custom_field_value==jjones"
```

HTTPIe Results

The updateapp.do call returns the appinfo XML document, which references the [appinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the [appinfo.xsd schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<appinfo xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;2.0&#x2f;appinfo"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;2.0&#x2f;appinfo

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
2.0&#x2f;appinfo.xsd" appinfo_version="1.1"
  account_id="<account id>">
  <application app_id="<app id>" app_name="<app name>"
description="<app description>"
  business_criticality="Very High" policy="Veracode Transitional
Very High"
  policy_updated_date="2019-08-13T14&#x3a;02&#x3a;08-04&#x3a;00"
teams="Demo Team" origin="Not Specified"
  industry_vertical="Other" app_type="Other"
deployment_method="Not Specified" is_web_application="false"
  archer_app_name="<archer app name>"
modified_date="2019-08-13T13&#x3a;59&#x3a;38-04&#x3a;00" cots="false"
  vast="false" business_unit="Not Specified" tags="">
    <customfield name="Custom 1" value=""/>
    <customfield name="Custom 2" value=""/>
    <customfield name="Custom 3" value=""/>
    <customfield name="Custom 4" value=""/>
    <customfield name="Custom 5" value=""/>
```

```

    <customfield name="Custom 6" value=""/>
    <customfield name="Custom 7" value=""/>
    <customfield name="Custom 8" value=""/>
    <customfield name="Custom 9" value=""/>
    <customfield name="Custom 10" value="jjones"/>
  </application>
</appinfo>

```

updatebuild.do

The `updatebuild.do` call updates the most recent build of an existing application in the portfolio. To update an earlier build, specify the `build_id` parameter.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/5.0/updatebuild.do>

Parameters

Name	Type	Description
<code>app_id</code>	Integer	Application ID.
Required		
<code>build_id</code>	Integer	Application or sandbox build ID.
<code>version</code>	String	Specify a unique identifier for the build.
<code>lifecycle_stage</code>	String	Validates against the names of the <code>Lifecycle</code> enums.
<code>launch_date</code>	String	Validates against the <code>mm/dd/yyyy</code> date format.
<code>sandbox_id</code>	Integer	Enter the ID of the sandbox that contains the build to update.

HTTPIE Example

Examples use the HTTPie command-line tool. See [Using HTTPie with the Python Authentication Library](#) on page 16.

```

http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/updatebuild.do" "app_id==<app
id>" "version==outdated1"

```

HTTPIE Results

The `updatebuild.do` call returns the `buildinfo` XML document, which references the [buildinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `buildinfo.xsd` [schema documentation](#).

```

<?xml version="1.0" encoding="UTF-8"?>

<buildinfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```

```

2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;4.0&#x2f;buildinfo"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;4.0&#x2f;buildinfo

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
4.0&#x2f;buildinfo.xsd" buildinfo_version="1.4"
  account_id="<account id>" app_id="<app id>" build_id="<build
id>">
    <build version="outdated1" build_id="<build id>"
      submitter="Veracode" platform="Not Specified"
      lifecycle_stage="Not Specified" results_ready="false"
      policy_name="Veracode Transitional Very High"
      policy_version="1" policy_compliance_status="Not Assessed"
      policy_updated_date="2019-08-13T14&#x3a;02&#x3a;08-04&#x3a;00"
      rules_status="Not Assessed" grace_period_expired="false"
      scan_overdue="false" legacy_scan_engine="false">
      <analysis_unit analysis_type="Static" status="Scan In Process"
      engine_version="20190805180615"/>
    </build>
  </buildinfo>

```

uploadfile.do

The `uploadfile.do` call uploads a file to an existing build or creates a build. Veracode recommends that you use the [uploadlargefile.do](#) call to avoid timeout errors when uploading a large file.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Before uploading additional files, ensure that:

- An upload or prescan is not in progress.
- The [beginscan.do](#) call is not in progress.
- If you recently ran the `beginscan.do` call, you did not set `auto_scan` to true.

Since the `uploadfile.do` call creates a build, if one does not already exist or if the most recent build has a published static scan, you do not need to call [createbuild.do](#). If the call creates a build, the build name is the date of the build with the scan type. For example, 03 Mar 2019 Static.


If you want to upload a file that does not have the same name as a previous file, you can use the `save_as` parameter to change the name, enabling flaw-matching with previously scanned files.

Resource URL

<https://analysiscenter.veracode.com/api/5.0/uploadfile.do>

Parameters

Name	Type	Description
app_id	Integer	Application ID.

Name	Type	Description
Required		
file		File to upload. The maximum file size is 2GB.
Required		<p>Requirements:</p> <ul style="list-style-type: none"> • Set Content-Type: multipart/form-data. • Open the file in binary mode. <p>If you see timeout errors during the upload, you can use the <code>call</code>.</p> <p> NOTE: You must enter the @ symbol before the entire specific filename.</p>
sandbox_id	Integer	Enter the ID of the target sandbox for the upload file.
save_as	String	Enter a new, unique filename for the uploaded file. The filename cannot contain slashes or periods.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -f "https://
analysiscenter.veracode.com/api/5.0/uploadfile.do" "app_id=<app id>"
"file@c:\myappfiles\myappzip.zip" "save_as=myappfile.zip"
```

HTTPIe Results

The `uploadfile.do` call returns the `filelist` XML document, which references the [filelist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the [filelist.xsd schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<filelist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/2.0/filelist"

xsi:schemaLocation="https://analysiscenter.veracode.com
/schema/2.0/filelist

https://analysiscenter.veracode.com/resource/2.0/filelist.xsd" filelist_version="1.1"
account_id="<account id>" app_id="<app id>" build_id="<build
id>">
  <file file_id="<file id>" file_name="myappfile.zip"
file_status="Uploaded" file_md5="<file md5>/>
```



```
<file file_id=<file id> file_name="myappfile2.zip"
file_status="Uploaded" file_md5=<file md5>"/>
</filelist>
```

Java Example

```
java -jar vosp-api-wrappers-java-<version #>.jar -vid <VeracodeApiId>
-vkey <VeracodeApiKey> -action uploadfile -appid <app id> -filepath c:
\Users\<username>\<filename>
```

Java Results

The `uploadfile.do` call returns the `filelist` XML document, which references the [filelist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the [filelist.xsd schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<filelist xmlns="https://analysiscenter.veracode.com/schema/2.0/
filelist"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  account_id=<account id> app_id=<app id> build_id=<build id>
filelist_version="1.1"
  xsi:schemaLocation="https://analysiscenter.veracode.com/
schema/2.0/filelist
  https://analysiscenter.veracode.com/resource/2.0/filelist.xsd">
  <file file_id=<file id> file_name="<file name>"
file_status="Uploaded"/>
</filelist>
```

uploadlargefile.do

The `uploadlargefile.do` call uploads a single file as a set of parts to an existing build or creates a build. Uploading the file in parts avoids timeout errors, which can occur when uploading a large file using the [uploadfile.do](#) call.

Veracode recommends using this call as an alternative to the `uploadfile.do` call. Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Before uploading additional files, ensure that:

- An upload or prescan is not in progress.
- The [beginscan.do](#) call is not in progress.
- If you recently ran the `beginscan.do` call, you did not set `auto_scan` to true.


Since the `uploadlargefile.do` call creates a build, if one does not already exist or if the most recent build has a published static scan, you do not need to call [createbuild.do](#). If the call creates a build, the build name is the date of the build with the scan type. For example, 03 Mar 2019 Static.

If you want to upload a file that does not have the same name as a previous file, you can use the `filename` parameter to change the name, enabling flaw-matching with previously scanned files.

Resource URL

`https://analysiscenter.veracode.com/api/5.0/uploadlargefile.do`

Parameters

Name	Type	Description
app_id Required	Integer	Application ID.
file Required		File to upload. The maximum file size is 2GB. Requirements: <ul style="list-style-type: none"> Set Content-Length: <number of bytes in the file> Set Content-Type: binary/octet-stream. <div>  NOTE: You must enter the @ symbol before the entire filename. </div>
filename	String	Enter a new, unique filename for the uploaded file. The filename cannot contain slashes or periods.
sandbox_id	Integer	Enter the ID of the target sandbox for the upload file.

Java Example

```
java -jar vosp-api-wrappers-java-<version #>.jar -vid <VeracodeApiId>
-vkey <VeracodeApiKey> -action uploadfile -appid <app id> -filepath c:
\Users\<username>\<filename>
```

Java Results

The `uploadlargefile.do` call returns the `filelist` XML document, which references the [filelist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `filelist.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<filelist xmlns="https://analysiscenter.veracode.com/schema/2.0/
filelist"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  account_id=<account id> app_id=<app id> build_id=<build id>
filelist_version="1.1"
  xsi:schemaLocation="https://analysiscenter.veracode.com/
schema/2.0/filelist
  https://analysiscenter.veracode.com/resource/2.0/filelist.xsd">
  <file file_id=<file id> file_name="<filename>"
file_status="Uploaded"/>
</filelist>
```

API Prescan Status Information

The prescan results that the Veracode XML APIs return contain several types of information in the XML documents. This table helps you understand the kind of information you may see.

The `prescanresults.xml` is returned when you call `getprescanresults.do`. The three types of information provide feedback about any issues with files and modules that you have uploaded.

File Issues	<p>Provides information about the files you have uploaded. If the scan engine fails to prescan the file, it returns a fatal error. Additional information that may be provided is in the element <code>file_issue</code> and attribute <code>details</code>. Each issues states whether it is (Required) or (Optional) to fix before you can run the application scan. Examples of returned message text include:</p> <ul style="list-style-type: none"> • Compiled without debug symbols. • Has corrupt headers. • Compiled using obfuscation.
Issue Details	<p>Provides you with general information about the prescan. If the prescan failed, you receive a fatal error. Additional information that may be provided is in the <code>module</code> element, as the child element <code>issue</code> and attribute <code>details</code>. Examples of these details include:</p> <ul style="list-style-type: none"> • No supporting files or PDB files • One or more uploaded files are compiled with a compiler that is not currently supported by Veracode. • One or more PHP files could not be compiled.
Module Status	<p>Provides you with general information about the modules that the scan engine has prescanned. If the module upload fails, the results you receive begins with (Fatal) and includes the <code>has_fatal_errors="true"</code> parameter. Additional information that may be provided is in the element <code>module</code> and attribute <code>status</code>. Examples of returned message text include:</p> <ul style="list-style-type: none"> • JSP Compilation Errors • PDB Files Not Loadable • Unsupported Architecture/Platform/Compiler

```

<prescanresults xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="https://analysiscenter.veracode.com/schema/2.0/prescanresults"
xsi:schemaLocation="<https://analysiscenter.veracode.com/schema/2.0/prescanresults
https://analysiscenter.veracode.com/resource/2.0/prescanresults.xsd>"
account_id="16069" app_id="35146" build_id="51163">

<module id="24549638" name="App_Browsers.dll" platform="CIL / CLR 2.0 / MSIL_MSVC8
_X86" size="4MB" status="Supporting Files Compiled without Debug Symbols - 1 File,
PDB Files Missing - 3 Files, PDB Files Not Loadable - 1 File, Missing Supporting
Files - 2 Files, Unsupported Framework - 1 File" has_fatal_errors="false">
<issue_details="Unsupported framework: .NET Remoting"/>
<file_issue filename="antlr.runtime.dll" details="Compiled without debug symbols
(Optional)"/>
<file_issue filename="App_web_register.aspx.5f83eb8c.pdb" details="Found
(Optional)"/>
<file_issue filename="App_web_javascriptflair.aspx.25fc4626.dll" details="Found
(Optional)"/>
<file_issue filename="system.data.sqlite.dll" details="Not Found (Optional)"/>
<file_issue filename="DotNetOpenAuth.pdb" details="Not Loadable (Optional)"/>
</module>

<module id="24549639" name="DotNetOpenAuth.dll" platform="CIL / CLR 2.0 / MSIL_MSVC8
_X86" size="1MB" status="PDB Files Missing - 1 File, PDB Files Not Loadable - 1
File, Unsupported Framework - 1 File" has_fatal_errors="false">
<issue_details="Unsupported framework: windows Communication Foundation, ASP.NET
MVC, .NET Web Routing, .NET Remoting"/>
<file_issue filename="system.web.Mvc.dll" details="Found (Optional)"/>
<file_issue filename="DotNetOpenAuth.pdb" details="Not Loadable (Optional)"/>
</module>

</prescanresults>

```

API Build Status Information

API users can verify the status of their scans by checking the returned XML. This table provides information on the status of builds that Veracode is currently scanning.

When a scan is in any of the states in this table, you are not allowed to upload new files to the application, nor are you allowed to add a new build. The only exception is when the scan is in the state of Results Ready and the scan results are published; at this time you can add a new build or upload files to the application.

State	Possible Action	New Build Allowed?	File Upload Allowed?
Incomplete	Build actions possible.	✗	✗
Not Submitted to Engine	Build actions possible.	✗	✗
Submitted to Engine	None - scan is processing.	✗	✗
Scan in Progress	None - scan is processing.	✗	✗
Scan Canceled	None - scan is processing.	✗	✗

State	Possible Action	New Build Allowed?	File Upload Allowed?
Pending Internal Review	None - scan is processing.	✗	✗
Results Ready	You can now create a new build.	✓	✓
Pre-scan Submitted	None - scan is processing.	✗	✗
Pre-scan Failed	You can upload your files again and run the prescan again. You cannot submit the scan until the prescan succeeds.	✓	✗
Pre-scan Success	Build actions possible.	✗	✗
No Modules Defined	You can upload your files again and run the prescan again. You cannot submit the scan until the prescan succeeds.	✗	✗
Pre-scan Canceled	Any action is possible.	✗	✗
Pending Vendor Confirmation	None - Veracode is waiting for the vendor.	✗	✗
Scan on Hold	Dynamic only - scan is processing.	✗	✗
Vendor Reviewing	None - scan is processing (enterprise)	✗	✗
	You can now create a new build (vendor).	✓	✓

API Tutorial: How to Scan an Application

This tutorial provides basic step-by-step information on how to use the Veracode Upload API to automate the scanning of an application using the HTTPie command-line tool. This guide uses

standalone HTTP request calls, but you can combine them in an API wrapper to process multiple API calls.



NOTE: Before starting with the APIs, ensure you have the [correct permissions](#) to use the APIs. Your Veracode user account must have API permissions to be able to access and use the APIs.

To configure and submit a scan request:

- 1 If your application already exists, omit this step. Create an application profile for the application you want to scan by entering:

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/createapp.do" "app_name==<your
application name>" "business_criticality==<enter level>
```

Where indicated, insert your application name and level of business criticality of the application. Refer to the [createapp.do](#) call for more information on these parameters. The returned `appinfo.xml` file contains the application ID number, which you need when using other calls.

- 2 Upload the file you want to scan by entering:

```
http --auth-type=veracode_hmac -f "https://
analysiscenter.veracode.com/api/5.0/uploadfile.do" "app_id==<your
app id>" "file@<your path and file name>" "save_as==<new name for
your app file>"
```

Where indicated, insert your application ID, and filename. Optionally, use the `save_as` parameter to give your application file a new name on the Veracode Platform.



NOTE: For the `file` parameter, enter the `@` symbol first followed by the path and filename.

Optionally, you can call [createbuild.do](#) if you want to name the scan.

- 3 Start the prescan of the uploaded file by entering:

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/beginprescan.do"
"app_id==<your application ID>"
```

Where indicated, insert your application ID.

- 4 Access the prescan results to know if it succeeded, allowing you to run the full scan. At this point you can add additional files using `uploadfile.do`, if necessary, but you can only do this if you have not set `auto_scan` to true as part of the `beginprescan.do` call. To start the scan, from the command line, enter:

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getprescanresults.do"
"app_id==<your application ID>"
```

Where indicated, insert your application ID. The returned `prescanresults.xml` document contains the prescan details. For more information about the prescan results, go to [API](#)

[Prescan Status Information](#). For more information on build status messages, see [API Build Status Information](#).

- 5 If your prescan was successful, start the full scan by entering:

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/beginscan.do" "app_id==<your
application ID>" -F "scan_all_top_level_modules==true"
```

Where indicated, insert your application ID.

Results APIs

Using the Results API

You can use the Veracode Results API to access your application assessment data from another application or a script.

For example, you can write a script to automatically download newly published scan results into a bug tracking system. You can also use this API to retrieve information in XML about application profiles, completed and in progress builds, and detailed application results data (including call stacks) or summary results data.

The Veracode Results API is a basic HTTPS-based request API that uses simple HTTP calls and returns data in XML format. You can use any technology that supports making HTTP calls and parsing XML to access the API.

To learn how to use the Results API, read the [tutorial](#).

For performance reasons, the Results API automatically compresses large XML files using Gzip if your requesting tool supports it. You are strongly encouraged to use a user-agent that supports Gzip to access the Results API. You can use any tool that supports HTTP to test the Results API.

Prerequisites

Before using the Results API, you must meet the following prerequisites:

- Either a Veracode non-human [API user account](#) with the Results API role, or a Veracode [human user account](#) with the Reviewer or Security Lead role.
- [Veracode API ID and key credentials](#)

Results API Calls

The Results API has several calls that automate tasks.

The following Results API calls are available for automating tasks that enable you to access your application assessment data in XML using another tool or script. Use these calls in your HTTP request.

API Call	Description
detailedreport.do	Compiles a detailed report for the specified application build.
detailedreportpdf.do	Produces a PDF file of the detailed report for the application build.

API Call	Description
getaccountcustomfieldlist.do	Returns the ID and name of the available custom fields in use by the organization of the user.
getappbuilds.do	Compiles a detailed list of applications and statuses, including all application and build profile data.
getcallstacks.do	Retrieves the call stacks for a static flaw.
summaryreport.do	Compiles a summary XML report for the specified application build.
summaryreportpdf.do	Produces a PDF file of the summary report for the specified application build.
thirdpartyreportpdf.do	Produces a PDF file of the scan results for a third-party application.

Mapping Results API Tasks

The Results API automates scan submissions and report retrieval.

The following table describes:

- How the Results API works
- When to use each API call
- The XML or PDF file an API returns
- How to map the API calls to the manual Veracode Platform processes

Step	Using the Veracode Platform	Using the Results API	Returns XML File
1	Go to the Application Overview for the chosen application.	getappbuilds.do , getapplist.do , getbuildlist.do	appbuilds.xml , applist.xml , buildlist.xml
2	View the Triage Flaws View online.	detailedreport.do , getcallstacks.do	detailedreport.xml , callstacks*.xml
3	View the Detailed Report online.	detailedreportpdf.do	detailedreport*.pdf
4	View the Summary Report online.	summaryreportpdf.do , thirdpartyreportpdf.do	summaryreport*.pdf , thirdpartyreport*.pdf

detailedreport.do

The `detailedreport.do` call returns a detailed XML report of all scan results related to the specified build.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/5.0/detailedreport.do>

Parameters

Name	Type	Description
build_id	Integer	Application or sandbox build ID.
Required		



NOTE: This call returns detailed flaw data only available for internally developed applications. Using this call for a third-party application returns an error.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/detailedreport.do"
"build_id=<build id>"
```

HTTPIe Results

The `detailedreport.do` call returns the `detailedreport` XML document, which references the [detailedreport.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `detailedreport.xsd` [schema documentation](#).

Partial example of the returned XML:

```
<?xml version='1.0' encoding='UTF-8'?>
  <detailedreport xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xmlns="https://www.veracode.com/schema/reports/export/1.0"
    xsi:schemaLocation="https://www.veracode.com/schema/reports/
export/1.0
https://analysiscenter.veracode.com/resource/
detailedreport.xsd"
    report_format_version="1.5" account_id="<account id>"
app_name="<app name>"
    app_id="<app id>" analysis_id="4705951"
static_analysis_unit_id="4721671"
    sandbox_id="<sandbox id>"
first_build_submitted_date="2019-08-13 17:57:41 UTC"
    version="13 Aug 2019 Static" build_id="4722565"
submitter="Veracode"
    platform="Not Specified" assurance_level="5"
business_criticality="5"
    generation_date="2019-09-03 19:54:36 UTC"
veracode_level="VL1" total_flaws="22"
    flaws_not_mitigated="22" teams="Demo Team"
life_cycle_stage="Not Specified"
    planned_deployment_date="" last_update_time="2019-08-13
18:08:47 UTC"
    is_latest_build="true" policy_name="Veracode Transitional
Very High"
    policy_version="1" policy_compliance_status="Did Not Pass"
```

```

        policy_rules_status="Did Not Pass"
    grace_period_expired="true"
        scan_overdue="false" business_owner="" business_unit="Not
Specified" tags=""
        legacy_scan_engine="false">
        <static-analysis rating="D" score="82"
submitted_date="2019-08-13 17:57:39 UTC"
        published_date="2019-08-13 18:08:35 UTC" version="13 Aug
2019 Static"
        analysis_size_bytes="16157840"
engine_version="20190805180615">
        <modules>
        <module name="httpd" compiler="GCC_Linux_IA32_3_4_6"
os="Red Hat Enterprise Linux v4 (IA32)" architecture="IA32"
loc="66813"
        score="82" numflawssev0="0" numflawssev1="0"
numflawssev2="6" numflawssev3="13"
        numflawssev4="0" numflawssev5="3" />
        </modules>
        </static-analysis>
        <severity level="5">
        <category categoryid="3" categoryname="Buffer Overflow"
pcirelated="false">
        <desc><para text="Buffer overflows (or buffer overruns)
occur
        when a program attempts to put more data in a buffer than
it has been allocated to hold.
        Writing to areas of memory not intended by the application
developer can lead to serious
        security vulnerabilities and can cause an application to
execute arbitrary code on behalf
        of an attacker." /><para text="The degree of exploitability
of buffer overflows varies
        depending on a number of factors, including buffer
location, execution path, and platform. Often, the resultant
behavior is limited
        to corrupting data or crashing the application. However,
in many cases, specially crafted attacks can be constructed that
        will execute arbitrary code with the privileges of the
vulnerable application. " />
        </desc>
        <recommendations><para text="There are a number of
mitigations that can be applied during both design and implementation
to
        prevent buffer overflows from occurring. Using multiple
techniques provides defense-in-depth. ">
        <bulletitem text="Always use bounded rather than unbounded
string manipulation functions, e.g. strncpy() and strncat()
instead of strcpy() and strcat()." />
        <bulletitem text="When using functions that accept a number
of bytes to copy, such as strncpy(), be aware that if the
        destination buffer size is equal to the source buffer size,
it may not null-terminate the string." />
        <bulletitem text="Be careful when working with multi-byte
strings, as the number of logical characters in a string is not
        equivalent to the number of bytes allocated in memory." /
><bulletitem text="Use a safe string handling functions such as
        Microsoft's strsafe.h. These functions prevent data from
being written past the end of buffers and guarantees null
        termination. Alternatively, use a string abstraction
library such as SafeStr, which automatically resizes strings as

```

```

        required. While neither of these approaches is foolproof,
        they will prevent many common mistakes." /></para>
    </recommendations>
    ...

```

API Wrapper Examples

Java example:

```

java -jar VeracodeJavaAPI.jar -vid <Veracode API ID> -vkey <Veracode
API Key> -action detailedreport -buildid <build id> -outputfilepath c:
\javawrappers\detailedreport.xml

```

C# example:

```

VeracodeC#API -vid <Veracode API ID> -vkey <Veracode API key> -action
detailedreport -buildid <build id> -outputfilepath c:\csharpwrappers
\detailedreport.xml

```

API Wrapper Results

The `detailedreport.do` call returns the `detailedreport` XML document, which references the [detailedreport.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `detailedreport.xsd` [schema documentation](#).

Partial example of XML output file:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
  <detailedreport xmlns="https://www.veracode.com/schema/
reports/export/1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    account_id="<account id>" analysis_id="4705951"
    app_id="<app id>" app_name="Apache" assurance_level="5"
    build_id="<build id>" business_criticality="5"
    business_owner="" business_unit="Not Specified"
    first_build_submitted_date="2019-08-13 17:57:41 UTC"
    flaws_not_mitigated="22" generation_date="2019-10-09 21:31:49
UTC"
    grace_period_expired="true" is_latest_build="true"
    last_update_time="2019-08-13 18:08:47 UTC"
    legacy_scan_engine="false" life_cycle_stage="Not Specified"
    planned_deployment_date=""
    platform="Not Specified" policy_compliance_status="Did Not
Pass"
    policy_name="Veracode Transitional Very High"
    policy_rules_status="Did Not Pass" policy_version="1"
    report_format_version="1.5" sandbox_id="1358509"
    scan_overdue="false" static_analysis_unit_id="4721671"
    submitter="Veracode" tags="" teams="Demo Team"
    total_flaws="22" veracode_level="VL1"
    version="13 Aug 2019 Static" xsi:schemaLocation="https://
www.veracode.com/schema/reports/export/1.0
https://analysiscenter.veracode.com/resource/
detailedreport.xsd">
    <static-analysis analysis_size_bytes="16157840"
    engine_version="20190805180615"

```

```

    published_date="2019-08-13 18:08:35 UTC" rating="D"
score="82" submitted_date="2019-08-13 17:57:39 UTC"
    version="13 Aug 2019 Static">
    <modules>
    <module architecture="IA32" compiler="GCC_Linux_IA32_3_4_6"
loc="66813" name="httpd" numflawssev0="0"
    numflawssev1="0" numflawssev2="6" numflawssev3="13"
numflawssev4="0" numflawssev5="3"
    os="Red Hat Enterprise Linux v4 (IA32)" score="82"/>
    </modules>
    </static-analysis>
    <severity level="5">
    <category categoryid="3" categoryname="Buffer Overflow"
pcirelated="false">
    <desc>
    <para text="Buffer overflows (or buffer overruns) occur when
a program attempts to put more data in a buffer
    than it has been allocated to hold. Writing to areas of
memory not intended by the application developer
    can lead to serious security vulnerabilities and can cause an
application to execute arbitrary code on
    behalf of an attacker."/>
    <para text="The degree of exploitability of buffer overflows
varies depending on a number of factors, including
    buffer location, execution path, and platform. Often, the
resultant behavior is limited to corrupting data
    or crashing the application. However, in many cases,
specially crafted attacks can be constructed that will
    execute arbitrary code with the privileges of the vulnerable
application. " />
    </desc>
    <recommendations>
    <para text="There are a number of mitigations that can be
applied during both design and implementation to
    prevent buffer overflows from occurring. Using multiple
techniques provides defense-in-depth. ">
    <bulletitem text="Always use bounded rather than unbounded
string manipulation functions, e.g. strncpy() and
    strncpyat() instead of strcpy() and strcat()."/>
    <bulletitem text="When using functions that accept a number
of bytes to copy, such as strncpy(), be aware that
    if the destination buffer size is equal to the source buffer
size, it may not null-terminate the string."/>
    <bulletitem text="Be careful when working with multi-byte
strings, as the number of logical characters in a
    string is not equivalent to the number of bytes allocated in
memory."/>
    <bulletitem text="Use a safe string handling functions such
as Microsoft's strsafe.h. These functions prevent
    data from being written past the end of buffers and
guarantees null termination. Alternatively, use a string
    abstraction library such as SafeStr, which automatically
resizes strings as required. While neither of these
    approaches is foolproof, they will prevent many common
mistakes."/>
    </para>
    </recommendations>

```

detailedreportpdf.do

The `detailedreportpdf.do` call downloads a PDF report of the detailed scan results for the specified build.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/4.0/detailedreportpdf.do>

Parameters

Name	Type	Description
build_id	Integer	Application or sandbox build ID.
Required		



NOTE: This call returns detailed flaw data only available for internally developed applications. Using this call for a third-party application returns an error.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -o detailedreport.pdf "https://
analysiscenter.veracode.com/api/4.0/detailedreportpdf.do"
"build_id==<build_id>"
```

HTTPIe Results

The `detailedreportpdf.do` call returns the `detailedreport_<app_name>_<build_id>` PDF format of the detailedreport XML file.

API Wrapper Examples

To get the same result as `detailedreportpdf.do` using an API wrapper, you use `-action detailedreport plus -format pdf`.

Java example:

```
java -jar VeracodeJavaAPI.jar -vid <Veracode API ID> -vkey <Veracode
API Key> -action detailedreport -buildid <build id> -format pdf -
outputfilepath c:\javawrappers\detailedreport.pdf
```

C# example:

```
VeracodeC#API -vid <Veracode API ID> -vkey <Veracode API key> -action
detailedreport -buildid <build id> -format pdf -outputfilepath c:
\csharpwrappers\detailedreport.pdf
```

API Wrapper Results

The `detailedreport` API wrapper call with `-format pdf` returns the `detailedreport_<app_name>_<build_id>` PDF format of the `detailedreport` XML file.

getaccountcustomfieldlist.do

The `getaccountcustomfieldlist.do` call returns the ID and name of the available custom fields for your organization. You must have the Results API role to be able to use this API.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/getaccountcustomfieldlist.do`

Parameters

This call takes no parameters.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getaccountcustomfieldlist.do"
```

HTTPIe Results

The `getaccountcustomfieldlist.do` call returns the `accountcustomfieldlist` XML document, which references the [accountcustomfieldlist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `accountcustomfieldlist.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

  <account_customfieldlist
xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;2001&#x2f;XMLSchema-
instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;accountcustomfieldlist"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;accountcustomfieldlist

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
accountcustomfieldlist.xsd"
  account_customfieldlist_version="1.0">
    <account_customfield id="132422" name="Custom 1"/>
    <account_customfield id="132423" name="Custom 2"/>
    <account_customfield id="132424" name="Custom 3"/>
    <account_customfield id="132425" name="Custom 4"/>
    <account_customfield id="132426" name="Custom 5"/>
```

```
<account_customfield id="132427" name="Custom 6"/>
<account_customfield id="132428" name="Custom 7"/>
<account_customfield id="132429" name="Custom 8"/>
<account_customfield id="132430" name="Custom 9"/>
<account_customfield id="132431" name="Custom 10"/>
</account_customfieldlist>
```



NOTE: There is no equivalent API wrapper call for `getaccountcustomfieldlist.do`.

getappbuilds.do

The `getappbuilds.do` call compiles a detailed list of applications and statuses, including all the application and scan profile data.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

The default is to include the latest published report for each application if it has changed in the last 31 days. This API returns data from the latest scan of an application. If you want to obtain more than just the data from the latest scan, use `getapplist.do` to generate a list of applications and `getbuildlist.do` to generate a list of scans for an application. You can then use `getappinfo.do` and `getbuildinfo.do` to retrieve the data about specific applications and scans.

Resource URL

<https://analysiscenter.veracode.com/api/4.0/getappbuilds.do>

Parameters

Name	Type	Description
<code>report_changed_since</code>	String	<p>Format: mm/dd/yyyy.</p> <p>Scan data is only included for scans of applications with reports published since the specified date. Changes to a report include:</p> <ul style="list-style-type: none"> Acceptance or rejection of a flaw mitigation relevant to the report Policy changes to the scan application profile since the report <p>The <code>only_latest</code> parameter also affects the data returned. The default is 31 days ago.</p>
<code>only_latest</code>	Boolean	<p>When true, the call returns the latest scan data for each application. Setting the parameter to false returns scan data for all previous scans. The <code>report_changed_since</code> parameter also affects the data returned. Default is true.</p>
<code>include_in_progress</code>	Boolean	<p>Setting this parameter to true includes scan data for all scans that have unpublished reports. To obtain data from non-published scans, set <code>include_in_progress</code> to true and <code>report_changed_since</code> to a date. Default is false.</p>

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/4.0/getappbuilds.do"
"report_changed_since==08/25/2019"
```

HTTPIe Results

The `getappbuilds.do` call returns the `appbuilds` XML document, which references the [applicationbuilds.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `applicationbuilds.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

    <applicationbuilds
xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;2001&#x2f;XMLSchema-
instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
#x2f;2.0&#x2f;applicationbuilds"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;2.0&#x2f;applicationbuilds

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
2.0&#x2f;applicationbuilds.xsd"
    account_id="<account id>"
    <application app_name="<app name>" app_id="<app id>"
industry_vertical="Manufacturing" assurance_level="Very High"
    business_criticality="Very High" origin="Not Specified"
modified_date="2019-08-13T14&#x3a;00&#x3a;10-04&#x3a;00"
    cots="false" business_unit="Not Specified" tags="">
    <customfield name="Custom 1" value=""/>
    <customfield name="Custom 2" value=""/>
    <customfield name="Custom 3" value=""/>
    <customfield name="Custom 4" value=""/>
    <customfield name="Custom 5" value=""/>
    <customfield name="Custom 6" value=""/>
    <customfield name="Custom 7" value=""/>
    <customfield name="Custom 8" value=""/>
    <customfield name="Custom 9" value=""/>
    <customfield name="Custom 10" value=""/>
    </application>
    </applicationbuilds>
    <!-- Parameters&#x3a; report_changed_since&#x3d;08&#x2f;
25&#x2f;2019 only_latest&#x3d;true include_in_progress&#x3d;false -->
```



NOTE: Use [getapplist.do](#) or [getbuildlist.do](#) to get scan information for multiple applications. If you use Archer, use the [Archer asynchronous APIs](#) to quickly obtain a detailed list of applications and statuses.

API Wrapper Examples

Java example:

```
java -jar VeracodeJavaAPI.jar -vid <Veracode API ID> -vkey <Veracode API Key> -action getappbuilds -reportchangedsince 08/25/2019
```

C# example:

```
VeracodeC#API -vid <Veracode API ID> -vkey <Veracode API key> -action getappbuilds -reportchangedsince 08/25/2019
```

API Wrapper Results

The `getappbuilds.do` call returns the `appbuilds` XML document, which references the [applicationbuilds.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `applicationbuilds.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
  <applicationbuilds xmlns="https://analysiscenter.veracode.com/
schema/2.0/applicationbuilds"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    account_id="74370"
    xsi:schemaLocation="https://analysiscenter.veracode.com/
schema/2.0/applicationbuilds
https://analysiscenter.veracode.com/resource/2.0/
applicationbuilds.xsd">
    <application app_id="<app id>" app_name="<app name>"
    assurance_level="Medium"
    business_criticality="Medium" business_owner="Veracode"
    business_unit="Mobile Secure Ventures" cots="false"
    industry_vertical="Not Specified"
    modified_date="2019-09-20T00:01:27-04:00" origin="Not
Specified" tags="">
      <customfield name="Custom 1" value=""/>
      <customfield name="Custom 2" value=""/>
      <customfield name="Custom 3" value=""/>
      <customfield name="Custom 4" value=""/>
      <customfield name="Custom 5" value=""/>
      <customfield name="Custom 6" value=""/>
      <customfield name="Custom 7" value=""/>
      <customfield name="Custom 8" value=""/>
      <customfield name="Custom 9" value=""/>
      <customfield name="Custom 10" value=""/>
      <build build_id="5003771" grace_period_expired="false"
    lifecycle_stage="Not Specified"
    platform="Not Specified" policy_compliance_status="Did Not
Pass"
    policy_name="Veracode Recommended Medium" policy_version="1"
    results_ready="true"
    rules_status="Did Not Pass" scan_overdue="false"
    submitter="<VeracodeUserName>"
    version="App Dynamic Scan">
      <analysis_unit analysis_type="Dynamic"
    published_date="2019-09-20T00:01:13-04:00"
    published_date_sec="1568952073" status="Results Ready"/>
    </build>
```

```

    </application>
    <application app_id="<app id>" app_name="<app name>"
assurance_level="Very High"
    business_criticality="Very High" business_unit="Not
Specified" cots="false"
    industry_vertical="Not Specified"
modified_date="2019-10-03T16:07:50-04:00"
    origin="Not Specified" tags="">
    <customfield name="Custom 1" value="" />
    <customfield name="Custom 2" value="" />
    <customfield name="Custom 3" value="" />
    <customfield name="Custom 4" value="" />
    <customfield name="Custom 5" value="" />
    <customfield name="Custom 6" value="" />
    <customfield name="Custom 7" value="" />
    <customfield name="Custom 8" value="" />
    <customfield name="Custom 9" value="" />
    <customfield name="Custom 10" value="" />
    </application>
</applicationbuilds><!-- Parameters&#x3a;
report_changed_since&#x3d;08&#x2f;25&#x2f;2019
only_latest&#x3d;true include_in_progress&#x3d;false -->

```

getcallstacks.do

The `getcallstacks.do` call retrieves the call stacks for a specified flaw in a specific build.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/5.0/getcallstacks.do>

Parameters

Name	Type	Description
build_id	Integer	Application or sandbox build ID.
Required		
flaw_id	Integer	To find flaw_id values, look for issueid fields in the details.
Required		



NOTE: This call returns detailed flaw data only available for internally developed applications. Using this call for a third-party application returns an error.

HTTPIE Example

Examples use the HTTPie command-line tool. See [Using HTTPie with the Python Authentication Library](#) on page 16.

```

http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getcallstacks.do"
"build_id=<build_id>" "flaw_id=13"

```

HTTPIe Results

The `getcallstacks.do` call returns the `callstacks_<app_id>_<build_id>_<flaw_id>` XML document, which references the [callstacks.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `callstacks.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<callstacks xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;2.0&#x2f;callstacks"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;2.0&#x2f;callstacks

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
2.0&#x2f;callstacks.xsd"
    callstacks_version="1.1" build_id="4722563" flaw_id="13">
  <callstack module_name="app_web_commentview.ascx.cd7a1e1e.dll"
steps="1" local_path="documents and settings&#x2f;
tjones&#x2f;my
documents&#x2f;demo&#x2f;blogenginedotnet&#x2f;
1.3&#x2f;blogengine.web&#x2f;themes&#x2f;
python-demo&#x2f;commentview.ascx"
function_name="__Render__controll" line_number="8">
  <call data_path="1" file_name="commentview.ascx"
file_path="documents and settings&#x2f;tjones&#x2f;my documents&#x2f;
demo&#x2f;blogenginedotnet&#x2f;
1.3&#x2f;blogengine.web&#x2f;themes&#x2f;python-
demo&#x2f;commentview.ascx"
function_name="__Render__controll" line_number="8"/>
  </callstack>
</callstacks>
```

API Wrapper Examples

Java example:

```
java -jar VeracodeJavaAPI.jar -vid <Veracode API ID> -vkey <Veracode
API Key> -action getcallstacks -buildid <build id> -flawid 13
```

C# example:

```
VeracodeC#API -vid <Veracode API ID> -vkey <Veracode API key> -action
getcallstacks -buildid <build id> -flawid 13
```

API Wrapper Results

The `getcallstacks.do` call returns the `callstacks_<app_id>_<build_id>_<flaw_id>` XML document, which references the [callstacks.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `callstacks.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

  <callstacks xmlns="https://analysiscenter.veracode.com/
schema/2.0/callstacks"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    build_id="<build id>"
    callstacks_version="1.1" flaw_id="13"
    xsi:schemaLocation="https://analysiscenter.veracode.com/
schema/2.0/callstacks
https://analysiscenter.veracode.com/resource/2.0/
callstacks.xsd">
    <callstack function_name="__Render__controll1" line_number="8"
      local_path="documents and settings/juno/my
documents/demo/blogengine.net/1.3/blogengine.web/themes/
barks/commentview.ascx"
      module_name="app_web_commentview.ascx.cd7alele.dll" steps="1">
      <call data_path="1" file_name="commentview.ascx"
file_path="documents and settings/juno/my
documents/demo/blogengine.net/1.3/blogengine.web/themes/
barks/commentview.ascx"
      function_name="__Render__controll1" line_number="8"/>
    </callstack>
  </callstacks>
```

summaryreport.do

The `summaryreport.do` call returns a summary XML report of the scan results for the specified build.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/4.0/summaryreport.do>

Parameters

Name	Type	Description
build_id	Integer	Application or sandbox build ID.
Required		



NOTE: This call returns detailed flaw data only available for internally developed applications. Using this call for a third-party application returns an error.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/4.0/summaryreport.do"
"build_id=<build id>"
```

HTTPIe Results

The `summaryreport.do` call returns the `summaryreport` XML document, which references the [summaryreport.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `summaryreport.xsd` [schema documentation](#).

```
<?xml version='1.0' encoding='UTF-8'?>

  <summaryreport xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xmlns="https://www.veracode.com/schema/reports/export/1.0"
    xsi:schemaLocation="https://www.veracode.com/schema/reports/
export/1.0
    https://analysiscenter.veracode.com/resource/
summaryreport.xsd" report_format_version="1.3"
    account_id="<account id>" app_name="<app name>"
    app_id="<app id>" analysis_id="4705951"
    static_analysis_unit_id="4721671"
      sandbox_id="1358509" first_build_submitted_date="2019-08-13
17:57:41 UTC" version="13 Aug 2019 Static" build_id="<build id>"
      submitter="Veracode" platform="Not Specified"
    assurance_level="5" business_criticality="5"
      generation_date="2019-09-04 20:06:49 UTC"
    veracode_level="VL1" total_flaws="22" flaws_not_mitigated="22"
    teams="Demo Team"
      life_cycle_stage="Not Specified" planned_deployment_date="
last_update_time="2019-08-13 18:08:47 UTC" is_latest_build="true"
      policy_name="Veracode Transitional Very High"
    policy_version="1" policy_compliance_status="Did Not Pass"
      policy_rules_status="Did Not Pass"
    grace_period_expired="true" scan_overdue="false" business_owner="
      business_unit="Not Specified" tags="
    legacy_scan_engine="false">
      <static-analysis rating="D" score="82"
submitted_date="2019-08-13 17:57:39 UTC" published_date="2019-08-13
18:08:35 UTC"
        version="13 Aug 2019 Static" analysis_size_bytes="16157840"
      engine_version="20190805180615">
        <modules>
          <module name="httpd" compiler="GCC_Linux_IA32_3_4_6"
os="Red Hat Enterprise Linux v4 (IA32)" architecture="IA32"
loc="66813"
            score="82" numflawssev0="0" numflawssev1="0"
numflawssev2="6" numflawssev3="13" numflawssev4="0"
numflawssev5="3" />
        </modules>
      </static-analysis>
      <severity level="5">
      <category categoryname="Numeric Errors" severity="Very
```

```

High" count="2" />
    <category categoryname="Buffer Overflow" severity="Very
High" count="1" />
    </severity>
    <severity level="4" />
    <severity level="3">
    <category categoryname="Buffer Management Errors"
severity="Medium" count="9" />
    <category categoryname="Numeric Errors" severity="Medium"
count="3" />
    <category categoryname="Cryptographic Issues"
severity="Medium" count="1" />
    </severity>
    <severity level="2">
    <category categoryname="Error Handling" severity="Low"
count="6" />
    </severity>
    <severity level="1" />
    <severity level="0" />
    <flaw-status new="22" reopen="0" open="0" cannot-
reproduce="0" fixed="0" total="22" not_mitigated="22" sev-1-
change="0" sev-2-change="6"
sev-3-change="13" sev-4-change="0" sev-5-change="3" />
    <customfields>
    <customfield name="Custom 1" value="" />
    <customfield name="Custom 2" value="" />
    <customfield name="Custom 3" value="" />
    <customfield name="Custom 4" value="" />
    <customfield name="Custom 5" value="" />
    <customfield name="Custom 6" value="" />
    <customfield name="Custom 7" value="" />
    <customfield name="Custom 8" value="" />
    <customfield name="Custom 9" value="" />
    <customfield name="Custom 10" value="" />
    </customfields>
    <software_composition_analysis third_party_components="0"
violate_policy="false" components_violated_policy="0">
    <vulnerable_components />
    </software_composition_analysis>
</summaryreport>

```

API Wrapper Examples

Java example:

```

java -jar vosp-api-wrappers-java-<version #>.jar -vid <Veracode API
ID> -vkey <Veracode API Key> -action summaryreport -buildid <build
id> -outputfilepath c:\javawrappers\summaryreport.xml

```

C# example:

```

VeracodeC#API -vid <Veracode API ID> -vkey <Veracode API key> -action
summaryreport -buildid <build id> -outputfilepath c:\csharpwrappers
\summaryreport.xml

```

API Wrapper Results

The `summaryreport.do` call returns the `summaryreport` XML document, which references the [summaryreport.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `summaryreport.xsd` [schema documentation](#).

A partial XML example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
  <summaryreport xmlns="https://www.veracode.com/schema/reports/
export/1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    account_id="<account id>" analysis_id="4978995"
    app_id="<app id>" app_name="<app name>" assurance_level="3"
    build_id="<build id>" business_criticality="3"
    business_owner="" business_unit="Mobile Secure Ventures"
    first_build_submitted_date="2019-09-18 21:15:28 UTC"
    flaws_not_mitigated="276" generation_date="2019-09-30
22:06:34 UTC" grace_period_expired="true" is_latest_build="true"
    last_update_time="2019-09-18 21:54:25 UTC"
    legacy_scan_engine="false" life_cycle_stage="Not Specified"
    planned_deployment_date="2019-09-18 21:12:23 UTC"
    platform="Not Specified" policy_compliance_status="Did Not Pass"
    policy_name="Veracode Recommended Medium"
    policy_rules_status="Did Not Pass" policy_version="1"
    report_format_version="1.3" sandbox_id="<sandbox id>"
    scan_overdue="false" static_analysis_unit_id="4994637"
    submitter="<Veracodeusername>" tags="" teams="Demo
Team,Release Team" total_flaws="276"
    veracode_level="VL1" version="18 Sep 2019 Static Promoted"
    xsi:schemaLocation="https://www.veracode.com/schema/reports/
export/1.0
https://analysiscenter.veracode.com/resource/
summaryreport.xsd">
    <static-analysis analysis_size_bytes="3735562"
    engine_version="20190826182718" next_scan_due="2019-12-18 22:54:12
UTC"
    published_date="2019-09-18 21:54:12 UTC" rating="C"
    score="53" submitted_date="2019-09-18 21:53:07 UTC"
    version="18 Sep 2019 Static Promoted">
      <modules>
        <module architecture="JVM" compiler="JAVAC_5" loc="40531"
name="<app name>" numflawssev0="1"
          numflawssev1="0" numflawssev2="19" numflawssev3="232"
numflawssev4="22" numflawssev5="2" os="Java J2SE 6"
          score="53"/>
        ...
```

summaryreportpdf.do

The `summaryreportpdf.do` call downloads a PDF summary report of the scan results for the specified build.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/4.0/summaryreportpdf.do`

Parameters

Name	Type	Description
build_id	Integer	Application or sandbox build ID.
Required		



NOTE: This call returns detailed flaw data only available for internally developed applications. Using this call for a third-party application returns an error.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -o summaryreport.pdf "https://
analysiscenter.veracode.com/api/4.0/summaryreportpdf.do"
"build_id=<build id>"
```

HTTPIe Results

The `summaryreportpdf.do` call returns the `summaryreport_<app_name>_<build_id>` PDF format of the `summaryreport` XML file.

API Wrapper Examples

To get the same result as `summaryreportpdf.do` using an API wrapper, you use the following parameters:

- `-action thirdpartyreport`
- `-format pdf`

Java example:

```
java -jar VeracodeJavaAPI.jar -vid <Veracode API ID> -vkey <Veracode
API Key> -action summaryreport -buildid <build id> -format pdf -
outputfilepath c:\javawrappers\summaryreport.pdf
```

C# example:

```
VeracodeC#API -vid <Veracode API ID> -vkey <Veracode API key> -action
detailedreport -buildid <build id> -format pdf -outputfilepath c:
\csharpwrappers\summaryreport.pdf
```

API Wrapper Results

The `summaryreportpdf.do` call returns the `summaryreport_<app_name>_<build_id>` PDF format of the `summaryreport` XML file.

thirdpartyreportpdf.do

The `thirdpartyreportpdf.do` call downloads a PDF file of the scan results for the specified build of a third-party application.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/4.0/thirdpartyreportpdf.do>

Parameters

Name	Type	Description
build_id	Integer	Application or sandbox build ID.
Required		

HTTPIE Example

Examples use the HTTPie command-line tool. See [Using HTTPie with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -o thirdpartyreport.pdf "https://  
analysiscenter.veracode.com/api/4.0/thirdpartyreportpdf.do"  
"build_id=<build id>"
```

HTTPIE Results

The `thirdpartyreportpdf.do` call returns the `thirdpartyreport_<app_name>_<build_id>` PDF format of the scan results of a third-party application.

API Wrapper Examples

To get the same result as `thirdpartyreportpdf.do` using an API wrapper, you use the following parameters:

- `-action thirdpartyreport`
- `-format pdf`

Java example:

```
java -jar VeracodeJavaAPI.jar -vid <Veracode API ID> -vkey <Veracode  
API Key> -action thirdpartyreport -buildid <build id> -format pdf -  
outputfilepath c:\javawrappers\thirdpartyreport.pdf
```

C# example:

```
VeracodeC#API -vid <Veracode API ID> -vkey <Veracode API Key> -action  
thirdpartyreport -buildid <build id> -format pdf -outputfilepath c:  
\csharpwrappers\thirdreport.pdf
```

API Wrapper Results

The `thirdpartyreportpdf.do` call returns the scan results of a third-party application as a PDF with filename `thirdpartyreport_<app_name>_<build_id>.pdf`.

API Tutorial: How to Access Scan Results

This tutorial provides basic step-by-step information on how to use the Veracode Results API to automate the retrieval of application scan results using the HTTPie command-line tool. This guide uses standalone HTTP request calls, but you can combine them in an API wrapper to process multiple API calls.



NOTE: Before starting with the APIs, ensure you have the correct permissions to use the APIs. Your Veracode user account must have [sufficient permissions](#) to access and use the APIs.

To retrieve detailed results for a specific application:

- 1 From the command-line, request the list of applications in your portfolio by entering the following:

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getapplist.do"
```

The returned `applist.xml` returns a list of application IDs and names, such as `app_id="18766" app_name="MyApp"`.

- 2 Obtain the list of builds for your chosen application.

- For policy scan results, enter the following command, using the application ID returned in the previous step:

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getbuildlist.do"
"app_id=<your application ID>"
```

The returned `buildlist.xml` from this step contains the IDs of the builds for this application.

- For sandbox scan results, enter the following command to obtain the IDs for your sandboxes, using the application ID returned in the previous step:

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getsandboxlist.do"
"app_id=<your application ID>"
```

When you have the ID for the chosen sandbox, enter the following command to obtain the build IDs for that sandbox:

```
http --auth-type=veracode_hmac "https://  
analysiscenter.veracode.com/api/5.0/getbuildlist.do"  
"app_id==<your application ID>" "sandbox_id==<your sandbox ID>"
```

3 To obtain the detailed report for your chosen build, enter:

```
http --auth-type=veracode_hmac "https://  
analysiscenter.veracode.com/api/5.0/detailedreport.do"  
"build_id==<the policy or sandbox build ID>"
```

Where indicated, insert the ID for the target application build or sandbox scan. Locate the build ID from the `buildlist.xml` or `sandboxlist.xml` returned in the previous step.

Admin APIs

Using the Admin API

The Veracode Admin API enables you to automate administration tasks such as creating and managing users and teams in the Veracode Platform and obtaining reference information about your organization.

For example, you can use a script to mass create users, mass assign users to teams, or deprovision many users at once. You can also use the Admin API to automatically connect Veracode to your organization's directory (LDAP or Active Directory), so that as changes occur in your directory (adding or terminating employees), those changes are automatically reflected in your user list in Veracode.

The Veracode Admin API is a basic HTTPS-based request API that uses simple HTTP calls and returns data in XML format. You can use any technology that supports making HTTP calls and parsing XML to access the API. For performance reasons, the Admin API automatically compresses large XML files using Gzip if your requesting tool supports it. You are strongly encouraged to use a user-agent that supports Gzip to access the Admin API. You can use any tool that supports HTTP to test the API.

Prerequisites

Before using the Admin API, you must meet the following prerequisites:

- A Veracode non-human [API user account](#) with the Admin API role, or a Veracode human user account with the Administrator role.
- [Veracode API ID and key credentials](#)

Admin API Calls

The Veracode Admin API uses several calls to automate user, team, and Veracode eLearning administration tasks.

Table 12: User Admin API Tasks

API Call	Description
<i>createuser.do</i>	Creates a new user login account. You cannot use this call to create a non-human <i>API user account</i> .
<i>deleteuser.do</i>	Deletes the specified human user account.
<i>getaccountcustomfieldlist.do</i>	Returns the ID and name of the available custom fields used by the organization of which the user is a member.
<i>getuserinfo.do</i>	Returns information about the specified user account.
<i>getuserlist.do</i>	Returns a list of all human Veracode user accounts for your organization.
<i>updateuser.do</i>	Changes or updates the information for the specified user account.

Table 13: Team Admin API Tasks

API Call	Description
<i>createteam.do</i>	Creates a new team of users.
<i>deleteteam.do</i>	Deletes the specified team of users.
<i>getteaminfo.do</i>	Returns detailed parameters of a specific team.
<i>getteamlist.do</i>	Returns a list of the user teams for your organization.
<i>updateteam.do</i>	Changes or updates the information of the specified user team.

Table 14: eLearning Admin API Tasks

API Call	Description
<i>getcurriculumlist.do</i>	Provides a list of the Veracode eLearning curricula defined for your organization.
<i>gettracklist.do</i>	Provides a list of the Veracode eLearning tracks available for your organization.

createuser.do

The `createuser.do` call creates a new human user account. You must be logged into a non-human user account to create a human user account with this call. To create a non-human API user account, you must create it in the Veracode Platform.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/3.0/createuser.do>

Permissions

Non-human user accounts require the [Admin API](#) role to use this call. Human user accounts require the [Administrator](#) role to use this call.

Parameters

Name	Type	Description
first_name	String	First name of the user.
Required		
last_name	String	Last name of the user.
Required		
email_address	String	Email address of the user.
Required		
custom_id	String	Required for SAML users only. The SAML Subject field value from account.
Required		
roles	String (case-sensitive)	<p>Comma-separated list of valid, human user roles:</p> <ul style="list-style-type: none"> • Administrator • Creator • Delete Scans • eLearning • Executive • Greenlight IDE User • Mitigation Approver • Policy Administrator • Reviewer • Sandbox Administrator • Sandbox User • Security Insights • Security Lead • Submitter • Vendor Manager <p>You cannot pass any of the non-human API user roles. If you specify Creator, Security Lead, or Submitter role, Veracode automatically applies the Any Scan permission to the scans.</p> <p>You can apply scan permissions to these types of scans: Static Analysis, DynamicDS, DynamicMP, Discovery, manual, and Dynamic Analysis.</p>
Required		
is_saml_user	Boolean	You can only update this parameter if your account is SAML-enabled.
login_enabled	Boolean	Specifies whether the user can log in to the account.
phone	String	Contact phone number for the user.
requires_token	Boolean	Specifies whether a user must provide a two-factor authentication to

Name	Type	Description
teams	String (case-sensitive)	Comma-separated list of team names. For a user with team membership restrictions, ensure all specified teams are on the Team Membership that user.
title	String	User title or position.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -o newuserinfo.xml "https://
analysiscenter.veracode.com/api/3.0/createuser.do"
"first_name==Regina" "last_name==Monarch"
"email_address==rmonarch@example.com" "teams==Demo Team"
"roles==Creator,Submitter,eLearning"
```

HTTPIe Results

The `createuser.do` call returns the `userinfo` XML document, which references the [userinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `userinfo.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<userinfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/userinfo/3.0"

xsi:schemaLocation="https://analysiscenter.veracode.com
&schema/userinfo/3.0"

https://analysiscenter.veracode.com/resource/3.0/userinfo.xsd" userinfo_version="3.0"
  username="rmonarch@example.com">
  <login_account first_name="Regina" last_name="Monarch"
login_account_type="user" email_address="rmonarch@example.com"
  login_enabled="true" requires_token="false" teams="Demo Team"
roles="Creator,eLearning,Submitter,Any Scan"
  is_elearning_manager="false" elearning_manager="No Manager"
elearning_track="No Track Assigned"
  elearning_curriculum="No Curriculum Assigned"
keep_elearning_active="false"/>
</userinfo>
```

API Wrapper Examples: Create Multiple Users

Using the `inputfilepath` parameter of the Java or C# API wrapper and a CSV file, you can make multiple calls to `createuser.do`.

The first column of the CSV contains the action (`createuser`) and each subsequent column represents a parameter. The first row of the CSV file contains the included parameter names. There must be a column for each required parameter. Columns for optional parameters may be omitted.

CSV data:

action	firstname	lastname	emailaddress	roles	teams
createuser	Wayne	Shorter	wayneshorter@example.com	Security Lead	Release Team
createuser	Tony	Williams	tonywilliams@example.com	"Creator,Submitter"	Demo Team
createuser	Carla	Bley	carlabley@example.com	Security Insights	

CSV file:

```
action,firstname,lastname,emailaddress,roles,teams
createuser,Wayne,Shorter,wayneshorter@example.com,Security
Lead,Release Team
createuser,Tony,Williams,tonywilliams@example.com,"Creator,Submitter",
Demo Team
createuser,Carla,Bley,carlabley@example.com,Security Insights
```

Java example:

```
java -jar VeracodeJavaAPI.jar -vid <Veracode API ID> -vkey <Veracode
API Key> -action createuser -inputfilepath c:\javawrappers
\newusers.csv
```

C# example:

```
VeracodeC#API -vid <Veracode API ID> -vkey <Veracode API key> -action
createuser -inputfilepath c:\csharpwrappers\newusers.csv
```

API Wrapper Results

The `createuser.do` call returns the `userinfo` XML document, which references the [userinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `userinfo.xsd` [schema documentation](#).

The API wrapper returns three XML documents:

```
<?xml version="1.0" encoding="utf-8"?>

<userinfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://analysiscenter.veracode.com/schema/userinfo/
  3.0"
  xsi:schemaLocation="https://analysiscenter.veracode.com/
  schema/userinfo/3.0
  https://analysiscenter.veracode.com/resource/3.0/
  userinfo.xsd" userinfo_version="3.0"
  username="wayneshorter@example.com">
```

```

    <login_account first_name="Wayne" last_name="Shorter"
login_account_type="user"
    email_address="wayneshorter@example.com"
login_enabled="true" requires_token="false"
    teams="Release Team" roles="Security Lead,Any Scan" />
</userinfo>

```

```

<?xml version="1.0" encoding="utf-8"?>

<userinfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="https://analysiscenter.veracode.com/schema/userinfo/
3.0"
    xsi:schemaLocation="https://analysiscenter.veracode.com/
schema/userinfo/3.0
    https://analysiscenter.veracode.com/resource/3.0/
userinfo.xsd" userinfo_version="3.0"
    username="tonywilliams@example.com">
    <login_account first_name="Tony" last_name="Williams"
login_account_type="user"
    email_address="tonywilliams@example.com"
login_enabled="true" requires_token="false"
    teams="Demo Team" roles="Creator,Submitter,Any Scan" />
</userinfo>

```

```

<?xml version="1.0" encoding="utf-8"?>

<userinfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="https://analysiscenter.veracode.com/schema/userinfo/
3.0"
    xsi:schemaLocation="https://analysiscenter.veracode.com/
schema/userinfo/3.0
    https://analysiscenter.veracode.com/resource/3.0/
userinfo.xsd" userinfo_version="3.0"
    username="carlabley@example.com">
    <login_account first_name="Carla" last_name="Bley"
login_account_type="user"
    email_address="carlabley@example.com" login_enabled="true"
requires_token="false"
    teams="" roles="Security Insights" />
</userinfo>

```

getuserinfo.do

The `getuserinfo.do` call returns information about the specified user.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/3.0/getuserinfo.do`

Permissions

Non-human user accounts require the [Admin API](#) role to use this call. Human user accounts require the [Administrator](#) role to use this call.

Parameters

Name	Type	Description
username	String	Required for non-SAML users. Usually the email address of the user.
Required		
custom_id	String	Required for SAML users. The SAML Subject field value from the SAML assertion.
Required		

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -o userinfo.xml "https://
analysiscenter.veracode.com/api/3.0/getuserinfo.do"
"username=tmonarch@example.com"
```

HTTPIe Results

The `getuserinfo.do` call returns the `userinfo` XML document, which references the [userinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `userinfo.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<userinfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/userinfo/3.0"

xsi:schemaLocation="https://analysiscenter.veracode.com
&#x2f;schema/userinfo/3.0"

https://analysiscenter.veracode.com/resource/3.0/userinfo.xsd" userinfo_version="3.0"
  username="tmonarch@example.com">
  <login_account first_name="Ted" last_name="Monarch"
login_account_type="user" email_address="tmonarch@example.com"
  login_enabled="true" requires_token="false" teams="Demo Team"
roles="Creator,eLearning,Submitter,Any Scan"
  is_elearning_manager="false" elearning_manager="No Manager"
elearning_track="No Track Assigned"
  elearning_curriculum="No Curriculum Assigned"
keep_elearning_active="false"/>
</userinfo>
```

getuserlist.do

The `getuserlist.do` call returns a list of the Veracode user accounts in your organization.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/3.0/getuserlist.do`

Permissions

Non-human user accounts require the [Admin API](#) role to use this call. Human user accounts require the [Administrator](#) role to use this call.

Parameters

There are no required parameters. Use the following parameters to filter the list:

Name	Type	Description
first_name	String	First name of the user.
last_name	String	Last name of the user.
custom_id	String	Required for SAML users. The SAML Subject field value from the SAML assertion.
email_address	String	Email address for the user.
login_account_type	String	Specify user or api.
phone	String	Contact phone number for the user.
teams	String	Comma-separated list of teams. This filter returns users matching the team names.
roles	String (case-sensitive)	Comma-separated list of roles. You can filter on these human roles: <ul style="list-style-type: none"> Administrator Creator eLearning Executive Mitigation Approver Policy Administrator Reviewer Security Insights Security Lead Submitter This filter only returns users that match all the listed roles. You can apply scan permissions to these types of scans: Static Analysis, DynamicMP, Discovery, manual, and Dynamic Analysis.
is_saml_user	Boolean	Specifies if the user has a SAML login.
login_enabled	Boolean	Specifies if the user can log in.
requires_token	Boolean	Specifies whether a user must provide a two-factor authentication token.
is_elearning_manager	Boolean	Specifies if the user is an Veracode eLearning manager.
elearning_manager	String	First and last name (not the username) of the Veracode eLearning manager. For example, Mary Doe.

Name	Type	Description
elearning_track	String	Veracode eLearning track name.
elearning_curriculum	String	Veracode eLearning curriculum name.
keep_elearning_active	Boolean	Specifies whether the Veracode eLearning subscription roll
custom_one	String	Custom field.
custom_two	String	Custom field.
custom_three	String	Custom field.
custom_four	String	Custom field.
custom_five	String	Custom field.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -o userlist.xml "https://
analysiscenter.veracode.com/api/3.0/getuserlist.do"
"login_enabled==true" "roles==Creator,Submitter"
```

HTTPIe Results

The `getuserlist.do` call returns a comma-separated list of usernames in the `userlist` XML document, which references the [userlist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `userlist.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<userlist xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;userlist&#x2f;3.0"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;userlist&#x2f;3.0

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
3.0&#x2f;userlist.xsd" userlist_version="3.0"
  account_id="<account id><filters/>
    <users
      usernames="aswallowtail&#x40;example.com,lpieris&#x40;example.com,svic
eroy&#x40;example.com,
      tmonarch&#x40;example.com,wbuckeye&#x40;example.com"/>
    </userlist>
```

updateuser.do

The `updateuser.do` call updates the information of the specified user account.

Use `updateuser.do` to change any of the parameters that you can change in the Veracode Platform, except for `login_account_type`.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/3.0/updateuser.do`

Permissions

Non-human user accounts require the [Admin API](#) role to use this call. Human user accounts require the [Administrator](#) role to use this call.

Parameters

Name	Type	Description
<code>username</code> Required	String	Required for non-SAML users only.
<code>custom_id</code> Required	String	Required for SAML users only. The SAML Subject field value. The <code>custom_id</code> is an identifier to inform the system which user you want to change the <code>custom_id</code> , use the <code>new_custom_id</code> .
<code>first_name</code>	String	First name of the user.
<code>last_name</code>	String	Last name of the user.
<code>email_address</code>	String	Valid email address. To change the username, send a new <code>username</code> parameter.
<code>phone</code>	String	Contact phone number for the user.
<code>teams</code>	String (case-sensitive)	Comma-separated list of team names.
<code>roles</code>	String (case-sensitive)	Comma-separated of these human user roles: <ul style="list-style-type: none"> Administrator Creator Delete Scans eLearning Executive Greenlight IDE User Mitigation Approver Policy Administrator Reviewer Sandbox Administrator Sandbox User Security Insights Security Lead

Name	Type	Description
		<ul style="list-style-type: none"> • Submitter • Vendor Manager <p>You cannot pass any of the non-human API user roles. If you pass the Security Lead, or Submitter role, Veracode automatically grants permission to the scans.</p> <p>You can apply scan permissions to these types of scans: Static Analysis, DynamicMP, Discovery, manual, and Dynamic Analysis.</p>
new_custom_id	String	For SAML users to change the identifier of the user account.
has_ip_restrictions	Boolean	Specifies whether the administrator has placed IP restrictions.
allowed_ip_addresses	String	A list of whitelisted IP addresses from which you can log in. This parameter is only used if <code>has_ip_restrictions</code> is true.
is_saml_user	Boolean	You can only update this parameter if your account is SAML. If you incorrectly sets <code>is_saml_user</code> to true, you receive an error. You must have SAML enabled and have a <code>custom_id</code> . If the account is not SAML but lacks a <code>custom_id</code> , you can set <code>is_saml_user</code> to true and provide a <code>custom_id</code> . You cannot set <code>is_saml_user</code> to false, because the account is converted to non-SAML users.
login_enabled	Boolean	Specifies whether the user can login.
requires_token	Boolean	Specifies whether a user must provide a two-factor authentication.
is_elearning_manager	Boolean	You can only update this parameter if your account has an active Veracode subscription.
elearning_manager	String	The first and last name (not the username) of the Veracode eLearning manager. For example, Mary Doe. You can only update this parameter if your account has an active Veracode eLearning subscription.
elearning_track	String	The Veracode eLearning track name. You can only update this parameter if your account has an active Veracode eLearning subscription.
elearning_curriculum	String	The Veracode eLearning curriculum name. You can only update this parameter if your account has an active Veracode eLearning subscription.
keep_elearning_active	Boolean	You can only update this parameter if your account has an active Veracode subscription.
custom_one	String	Custom field.
custom_two	String	Custom field.
custom_three	String	Custom field.
custom_four	String	Custom field.
custom_five	String	Custom field.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -o updateduserinfo.xml "https://
analysiscenter.veracode.com/api/3.0/updateuser.do"
"username==tmonarch@example.com" "phone==111-222-3333"
```

HTTPIe Results

The `updateuser.do` call returns the output XML document, which references the [userinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the [userlist.xsd schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<userinfo xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;userinfo&#x2f;3.0"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;userinfo&#x2f;3.0

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
3.0&#x2f;userinfo.xsd" userinfo_version="3.0"
  username="tmonarch&#x40;example.com">
  <login_account first_name="Ted" last_name="Monarch"
login_account_type="user" email_address="tmonarch&#x40;example.com"
  phone="111-222-3333" login_enabled="true"
requires_token="false" teams="Demo Team"
  roles="Creator,eLearning,Submitter,Any Scan"
is_elearning_manager="false" elearning_manager="No Manager"
  elearning_track="No Track Assigned" elearning_curriculum="No
Curriculum Assigned" keep_elearning_active="false"/>
</userinfo>
```

createteam.do

The `createteam.do` call creates a new team of users.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/3.0/createteam.do`

Permissions

Non-human user accounts require the [Admin API](#) role to use this call. Human user accounts require the [Administrator](#) role to use this call.

Parameters

Name	Type	Description
team_name	String	Must be unique.
Required		
members	String	Comma-separated list of usernames.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -o newteaminfo.xml "https://
analysiscenter.veracode.com/api/3.0/createteam.do"
"team_name==Release Team"
"members==bmetalmark@example.com,tmonarch@example.com"
```

HTTPIe Results

The createteam.do call returns the teaminfo XML document, which shows a user list filtered by the new team name, listing the users in the team. This XML document references the [teaminfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the teaminfo.xsd [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<teaminfo xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;teaminfo&#x2f;3.0"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;teaminfo&#x2f;3.0

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
3.0&#x2f;teaminfo.xsd" teaminfo_version="3.1"
  account_id="<account id>" team_id="<team id>"
  team_name="Release Team" creation_date="09&#x2f;06&#x2f;2019">
  <users
    usernames="bmetalmark&#x40;example.com,tmonarch&#x40;example.com"/>
  </teaminfo>
```

deleteteam.do

The deleteteam.do call deletes a specified team.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/3.0/deleteteam.do>

Permissions

Non-human user accounts require the [Admin API](#) role to use this call. Human user accounts require the [Administrator](#) role to use this call.

Parameters

Name	Type	Description
team_id	Integer	Team ID.
Required		

HTTPIE Example

Examples use the HTTPie command-line tool. See [Using HTTPie with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -o teamlist.xml "https://
analysiscenter.veracode.com/api/3.0/deleteteam.do" "team_id=145690"
```

HTTPIE Results

The deleteteam.do call returns the teamlist XML document, which references the [teamlist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the [teamlist.xsd schema documentation](#).

The return lists the remaining teams.

```
<?xml version="1.0" encoding="UTF-8"?>

<teamlist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/teamlist/3.0"

xsi:schemaLocation="https://analysiscenter.veracode.com
/schema/teamlist/3.0

https://analysiscenter.veracode.com/resource/teamlist.xsd" teamlist_version="3.0"
  account_id="<account id>"
    <team team_id="145696" team_name="Debug Team"
creation_date="09/06/2019"/>
    <team team_id="144659" team_name="Demo Team"
creation_date="08/13/2019"/>
    <team team_id="145675" team_name="Quality Team"
creation_date="09/06/2019"/>
    <team team_id="145689" team_name="Release Team"
creation_date="09/06/2019"/>
  </teamlist>
```

getteaminfo.do

The getteaminfo.do call returns information about a specific team.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/3.0/getteaminfo.do>

Permissions

Non-human user accounts require the [Admin API](#) role to use this call. Human user accounts require the [Administrator](#) role to use this call.

Parameters

Name	Type	Description
team_id	Integer	Team ID.
Required		
include_users	String	Specify <code>Yes</code> to view the members of the team. Default is <code>No</code> .
include_applications	String	Specify <code>Yes</code> to view applications assigned to the team. Default is <code>No</code> .

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -o teaminfo.xml "https://
analysiscenter.veracode.com/api/3.0/getteaminfo.do" "team_id==144659"
"include_users==yes" "include_applications==yes"
```

HTTPIe Results

The `getteaminfo.do` call returns the `teaminfo` XML document, which references the [teaminfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `teaminfo.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<teaminfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/teaminfo/3.0"

xsi:schemaLocation="https://analysiscenter.veracode.com
/schema/teaminfo/3.0"

https://analysiscenter.veracode.com/resource/teaminfo.xsd"
teaminfo_version="3.1" team_name="Demo Team"
can_view_shared_types="false" business_unit="<business unit>">
  <user username="sviceroy@x26;x23;x40;x3b;example.com"
```

```

first_name="Saiya" last_name="Viceroy"
  email_address="sviceroy&#x40;example.com"/>
  <user username="aswallowtail&#x26;&#x23;x40&#x3b;example.com"
first_name="Anna" last_name="Swallowtail"
  email_address="aswallowtail&#x40;example.com"/>
  <user username="tmonarch&#x26;&#x23;x40&#x3b;example.com"
first_name="Ted" last_name="Monarch"
  email_address="tmonarch&#x40;example.com"/>
  <user username="lpieris&#x26;&#x23;x40&#x3b;example.com"
first_name="Li" last_name="Pieris"
  email_address="lpieris&#x40;example.com"/>
  <application app_id="<app id>" app_name="<app name>"
assessment_type="SDLC" account_id="<account id>"
  business_unit="Not Specified"/>
  <application app_id="<app id>" app_name="<app name>"
assessment_type="SDLC" account_id="<account id>"
  business_unit="Not Specified"/>
</teaminfo>

```

getteamlist.do

The `getteamlist.do` call returns a list of the user teams in your organization.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/3.0/getteamlist.do`

Permissions

Non-human user accounts require the [Admin API](#) role to use this call. Human user accounts require the [Administrator](#) role to use this call.

Parameters

This call takes no parameters.

HTTPIE Example

Examples use the HTTPie command-line tool. See [Using HTTPie with the Python Authentication Library](#) on page 16.

```

http --auth-type=veracode_hmac -o teamlist.xml "https://
analysiscenter.veracode.com/api/3.0/getteamlist.do"

```

HTTPIE Results

The `getteamlist.do` call returns the `teamlist` XML document, which references the [teamlist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `teamlist.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<teamlist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/teamlist/3.0"

xsi:schemaLocation="https://analysiscenter.veracode.com/schema/teamlist/3.0

https://analysiscenter.veracode.com/resource/teamlist.xsd" teamlist_version="3.0"
  account_id="<account id>"
  <team team_id="145696" team_name="Debug Team"
creation_date="09/06/2019"/>
  <team team_id="144659" team_name="Demo Team"
creation_date="08/13/2019"/>
  <team team_id="145690" team_name="Engineering"
creation_date="09/06/2019"/>
  <team team_id="145675" team_name="Quality Team"
creation_date="09/06/2019"/>
  <team team_id="145689" team_name="Release Team"
creation_date="09/06/2019"/>
</teamlist>

```

updateteam.do

The updateteam.do call updates or changes the member information of the specified team.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/3.0/updateteam.do>

Permissions

Non-human user accounts require the [Admin API](#) role to use this call. Human user accounts require the [Administrator](#) role to use this call.

Parameters

Name	Type	Description
team_id	Integer	Team ID.
Required		
members	String	Comma-separated list of team member names. Provide the list to replace the current list.
team_name	String	To change the team name, provide a new name.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

To get the current membership of the team before you change it, call `getuserlist.do`.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/3.0/getuserlist.do" "teams==<teamid>"
```

This example updates the members of the team.

```
http --auth-type=veracode_hmac -o updatedteaminfo.xml "https://
analysiscenter.veracode.com/api/3.0/updateteam.do"
"members="aswallowtail@example.com,lpieris@example.com,sviceroy@exampl
e.com,tmonarch@example.com,wbuckeye@example.com" "team_id=144659"
```

HTTPie Results

The `updateteam.do` call returns the `teaminfo` XML document, which references the [teaminfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `teaminfo.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<teaminfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/teaminfo/3.0"

xsi:schemaLocation="https://analysiscenter.veracode.com
/schema/teaminfo/3.0

https://analysiscenter.veracode.com/resource/teaminfo.xsd" teaminfo_version="3.1"
  account_id="<account id>" team_id="144659" team_name="Demo
Team" creation_date="08/13/2019">
  <users
    usernames="lpieris@example.com,wbuckeye@example.com,tmonarch
@example.com"/>
</teaminfo>
```

getcurriculumlist.do

The `getcurriculumlist.do` call returns a list of the Veracode eLearning curricula defined for your organization.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/3.0/getcurriculumlist.do`

Permissions

Non-human user accounts require the [Admin API](#) role to use this call. Human user accounts cannot use this call.

Parameters

This call takes no parameters.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -o curriculumlist.xml "https://analysiscenter.veracode.com/api/3.0/getcurriculumlist.do"
```

HTTPIe Results

The `getcurriculumlist.do` call returns the `curriculumlist` XML document, which references the [curriculumlist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `curriculumlist.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<curriculumlist xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&#x2f;curriculumlist&#x2f;3.0"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&#x2f;curriculumlist&#x2f;3.0

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;3.0&#x2f;curriculumlist.xsd"

  curriculumlist_version="3.0" account_id="<account id>">
  <curriculums curriculum_names="Beginner Curriculum"/>
```

gettracklist.do

The `gettracklist.do` call returns a list of the Veracode eLearning tracks that are available for your organization.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/3.0/gettracklist.do`

Permissions

Non-human user accounts require the [Admin API](#) role to use this call. Human user accounts cannot use this call.

Parameters

This call takes no parameters.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -o tracklist.xml "https://
analysiscenter.veracode.com/api/3.0/gettracklist.do"
```

HTTPIe Results

The `gettracklist.do` call returns the `tracklist` XML document, which references the [tracklist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `tracklist.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<tracklist xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;tracklist&#x2f;3.0"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;tracklist&#x2f;3.0

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
3.0&#x2f;tracklist.xsd"
  tracklist_version="3.0" account_id="<account id>">
  <tracks track_names="eLearning,eLearning Awareness"/>
</tracklist>
```

getmaintenancescheduleinfo.do

The `getmaintenancescheduleinfo.do` call requests the upcoming Veracode Platform maintenance schedule. Plan to pause scheduled API activity during the downtime to avoid automation failures.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/3.0/`
`getmaintenancescheduleinfo.do`

Permissions

Non-human user accounts require the [Admin API](#) role to use this call. Human user accounts require the [Administrator](#) role to use this call.

Parameters

This call takes no parameters.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -o maintenancescheduleinfo.xml
"https://analysiscenter.veracode.com/api/3.0/
getmaintenancescheduleinfo.do"
```

HTTPIe Results

The `getmaintenancescheduleinfo.do` call returns the `maintenancescheduleinfo` XML document, which references the [maintenancescheduleinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `maintenancescheduleinfo.xsd` [schema documentation](#).

If the date in the XML return is in the past, Veracode has not yet scheduled the next maintenance window.

```
<?xml version="1.0" encoding="UTF-8"?>

<maintenancescheduleinfo
  xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;2001&#x2f;XMLSchema-
  instance"

  xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
  &#x2f;3.0&#x2f;maintenancescheduleinfo"

  xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
  &#x2f;schema&#x2f;3.0&#x2f;maintenancescheduleinfo

  https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
  3.0&#x2f;maintenancescheduleinfo.xsd">
  <maintenanceschedule>
    <downtime_start>2019-08-28T16&#x3a;44&#x3a;42-04&#x3a;00</
  downtime_start>
    <downtime_end>2019-08-28T16&#x3a;44&#x3a;42-04&#x3a;00</
  downtime_end>
    <title>Scheduled Maintenance Notification</title>
    <description>The Veracode service will be unavailable due to
  maintenance.&#xa;We apologize for any inconvenience caused
    during this period.&#xa; Please contact your primary
  services manager or Veracode Support
    &#x28;&#x3c;a
  href&#x3d;&#x22;mailto&#x3a;support&#x40;veracode.com&#x22;&#x3e;
  support&#x40;veracode.com&#x3c;&#x2f;a&#x3e;&#x29; if you
  have any questions.</description>
  </maintenanceschedule>
</maintenancescheduleinfo>
```

Mitigation APIs

Using the Mitigation and Comments API

The Mitigation and Comments API enables you to integrate comments on findings and mitigation workflow tasks into IDEs and bug tracking systems.

You can mitigate a finding, accept or reject a mitigation action, or comment on a proposed mitigation. In addition, you can view all comments and mitigation actions any user has performed on a finding.

To learn about how to use the Mitigation and Comments API, read the [tutorial](#).

Prerequisites

Before using the Mitigation and Comments API, you must meet the following prerequisites:

- A Veracode non-human [API user account](#) with the Mitigation API role, or a Veracode human user account with one of the following roles:

Reviewer or Security Lead To view all actions performed on a finding, to submit proposed mitigations, or to comment on proposed mitigations.

Mitigation Approver and either Reviewer or Security Lead To accept or reject proposed mitigations.

- [Veracode API ID and key credentials](#)

Mitigation and Comments API Calls

The following Mitigation and Comments API calls are available for automating tasks:

Table 15: Mitigation and Comments API Tasks

API Call	Description
getmitigationinfo.do	Retrieves all the information on actions that users have performed on a list of one or more flaws in the specified build.
updatemitigationinfo.do	Performs various actions on a discovered flaw, such as commenting on a flaw, proposing a mitigation action, and rejecting or accepting a mitigation action.

getmitigationinfo.do

The `getmitigationinfo.do` call retrieves all action information for listed flaws in the specified build.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/getmitigationinfo.do>

Parameters

Name	Type	Description
build_id	Integer	ID of the latest build.
Required		

Name	Type	Description
flaw_id_list	String	Comma-separated list of flaw IDs. Find flaw IDs on the Triage Veracode Platform or in the <code>issueid</code> fields returned by the
Required		

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/getmitigationinfo.do"
"build_id=<build id>" "flaw_id_list=2,3,7,20,25"
```

HTTPIe Results

The `getmitigationinfo.do` call returns the `mitigationinfo` XML document, which references [mitigationinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `mitigationinfo.xsd` [schema documentation](#).

```
<mitigationinfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="https://analysiscenter.veracode.com/schema/mitigationinfo/1.0"
xsi:schemaLocation="https://analysiscenter.veracode.com/schema/mitigationinfo/1.0
https://analysiscenter.veracode.com/resource/mitigationinfo.xsd" mitigationinfo_version="1.1"
  build_id="<build id>">
  <issue flaw_id="2" category="Exposure of Private Information
('Privacy Violation')">
    <mitigation_action action="appdesign" desc="Mitigate by Design"
reviewer="VendorTechnique : M1 : Establish and maintain
control over all of your inputs
Specifics : Specifics comment added by vendor.
Remaining Risk : Remaining Risk comment added by vendor.
Verification : Verification comment added by vendor."/>
    <mitigation_action action="deviates" desc="Deviates from
Guidelines" reviewer="Veracode" date="2017-01-20 02:29:32"
comment="Deviates comment added by internal admin."/>
  </issue>
  <issue flaw_id="3" category="Exposure of Private Information
('Privacy Violation')">
    <mitigation_action action="appdesign" desc="Mitigate by Design"
reviewer="VendorTechnique : M1 : Establish and maintain
control over all of your inputs
Specifics : Specifics comment added by vendor.
Remaining Risk : Remaining Risk comment added by vendor.
Verification : Verification comment added by vendor."/>
    <mitigation_action action="conforms" desc="Conforms to
Guidelines" reviewer="Veracode" date="2017-01-20 02:29:07"
comment="Conform comment added by internal admin."/>
  </issue>
```

```

    <issue flaw_id="20" category="Process Control"/>
    <issue flaw_id="25" category="Missing Release of Memory after
Effective Lifetime"/>
    <issue flaw_id="7" category="Authorization Bypass Through User-
Controlled Key">
      <mitigation_action action="appdesign" desc="Mitigate by Design"
reviewer="<Veracodeuserid>" date="2019-09-17 12&#x3a;16&#x3a;48"
comment="Test mitigation details."/>
    </issue>
  </mitigationinfo>

```

updatemitigationinfo.do

The `updatemitigationinfo.do` call enables you to perform several actions on a list of one or more flaws in the specified build. These actions include commenting on a flaw, proposing a mitigation action, and rejecting or accepting a mitigation action.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/updatemitigationinfo.do>

Parameters

Name	Type	Description
build_id Required	Integer	ID of the latest build, otherwise the call fails.
action Required	String	Valid values are: <ul style="list-style-type: none"> comment fp (false positive) appdesign osenv netenv rejected accepted
comment Required	String	Comment string to associate with the action. Limit of 2048
flaw_id_list Required	String	Comma-separated list of flaw IDs. Find flaw IDs on the Veracode Platform or in the <code>issueid</code> fields returned by the

HTTPIe Examples

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```

http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/updatemitigationinfo.do"

```

```
"build_id==<build id>" "action==comment" "comment==Test comment."
"flaw_id_list==5,7,49"

http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/updatemitigationinfo.do"
"build_id==<build id>" "action==appdesign" "comment==Test mitigation
details." "flaw_id_list==7"
```

HTTPIe Results

The `updatemitigationinfo.do` call returns the `mitigationinfo` XML document, which references the [mitigationinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `mitigationinfo.xsd` [schema documentation](#).

In this comment-only example, note that this API handles non-existent flaw IDs in an `error` element at the end of the XML.

```
<?xml version="1.0" encoding="UTF-8"?>

<mitigationinfo xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;mitigationinfo&#x2f;1.0"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;mitigationinfo&#x2f;1.0

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
mitigationinfo.xsd" mitigationinfo_version="1.1"
  build_id="<build id>">
  <issue flaw_id="7" category="Authorization Bypass Through User-
Controlled Key">
    <mitigation_action action="comment" desc="Add Comment"
reviewer="<Veracodeusername>" date="2019-09-16 10&#x3a;33&#x3a;00"
comment="Test comment."/>
  </issue>
  <issue flaw_id="5" category="Unchecked Error Condition">
    <mitigation_action action="comment" desc="Add Comment"
reviewer="<Veracodeusername>" date="2019-09-16 10&#x3a;33&#x3a;00"
comment="Test comment."/>
  </issue>
  <error type="not_found" flaw_id_list="49"/>
</mitigationinfo>
```

In the mitigated-by-design example, note that the API returns the complete list of actions for the flaw ID with the new action added to the end.

```
<?xml version="1.0" encoding="UTF-8"?>

<mitigationinfo xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;mitigationinfo&#x2f;1.0"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
```

```
&#x2f;schema&#x2f;mitigationinfo&#x2f;1.0

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
mitigationinfo.xsd" mitigationinfo_version="1.1"
  build_id="<build id>"
  <issue flaw_id="7" category="Authorization Bypass Through User-
Controlled Key">
    <mitigation_action action="comment" desc="Add Comment"
reviewer="<Veracodeuserid>" date="2019-09-16 10&#x3a;33&#x3a;41"
    comment="Test comment."/>
    <mitigation_action action="appdesign" desc="Mitigate by Design"
reviewer="<Veracodeuserid>" date="2019-09-17 12&#x3a;16&#x3a;48"
    comment="Test mitigation details."/>
  </issue>
</mitigationinfo>
```

API Tutorial: How to Use the Mitigation Calls

This tutorial provides basic information on some of the tasks the Mitigation and Comments API can do. This guide uses standalone HTTP request calls, but you can combine them in an API wrapper to process multiple API calls.



NOTE: Before starting with the APIs, ensure you have the correct permissions to use the APIs. Your Veracode user account must have sufficient permissions to access and use the APIs.

- 1 To mark a flaw found in scan results as a false positive, from the command line, enter:

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/updatemitigationinfo.do"
"build_id==<your build ID>" "action==fp" "comment==<your comment
text>" "flaw_id_list==<your flaw IDs>
```

Where required, enter the build ID, which you can get from the `buildlist.xml` returned by the [getbuildlist.do](#) call. Also, enter a comma-separated list of flaw IDs, which you find in the Triage Flaws page for that application in the Veracode Platform. You can also find the flaw IDs in the [detailedreport.xml](#) file.

To create a list of builds of your chosen application, enter:

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getbuildlist.do"
"app_id==<your application ID>"
```

Enter your application ID from the returned `applist.xml` from the previous step. The returned `buildlist.xml` from this step contains the IDs of the builds for the application, such as:

```
<buildlist>
  <build build_id="49894" version="5.0"/>
</buildlist>
```

2 To accept a flaw found in scan results, enter:

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/updatemitigationinfo.do"
"build_id==<your build id>" "action==accepted" "comment==<your
comment text>" "flaw_id_list==<your flaw IDs>"
```

Where required, enter the build ID and a comma-separated list of flaw IDs.

DynamicDS APIs

Using the DynamicDS API Calls

The Veracode DynamicDS APIs use several calls to automate tasks involved in creating and configuring DynamicDS requests.

These calls consist of:

<code>createdynamicscan.do</code>	Creates a DynamicDS scan request.
<code>dynamicscanconfig.do</code>	Configures a DynamicDS scan request.
<code>dynamicincludeexclude.do</code>	Configures a DynamicDS scan request to include and exclude specified URLs.
<code>submitdynamicscan.do</code>	Submits a DynamicDS scan request.
<code>addbrowserbasedlogin.do</code>	Provides username and password credentials for authentication of browser-based logins for a DynamicDS scan.
<code>uploadformsbasedloginscript.do</code>	Uploads a forms-based login script for a DynamicDS scan.
<code>getdynamicstatus.do</code>	Returns the status of a specific DynamicDS scan.
<code>getdynamicflaws.do</code>	Returns information on a specific flaw from a DynamicDS scan.
<code>getvsalist.do</code>	Returns a list of available VSAs.
<code>assignvsa.do</code>	Assigns a VSA to a DynamicDS scan request.
<code>rescandynamicscan.do</code>	Creates a DynamicDS scan request using the configuration of the last scan.

`createdynamicscan.do`

The `createdynamicscan.do` call creates a DynamicDS scan request.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

After calling `createdynamicscan.do`, call the scan configuration APIs as needed:

- [dynamicscanconfig.do](#)
- [dynamicincludeexclude.do](#)
- [addbrowserbasedlogin.do](#)
- [uploadformsbasedloginscript.do](#)

Then call [submitdynamicscan.do](#) to start the scan.

Resource URL

<https://analysiscenter.veracode.com/api/5.0/createdynamicscan.do>

Permissions

You need the [Upload and Scan API role](#) to use this call.

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		
scan_name	String	Unique identifier for the new scan.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/createdynamicscan.do"
"app_id==<app id>" "scan_name==Test DynamicDS Scan"
```

HTTPIe Results

The `createdynamicscan.do` call returns the `dynamic_scan_info` XML document, which references the [dynamicscaninfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `dynamicscaninfo.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<dynamic_scan_info xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/4.0/dynamicscaninfo"

xsi:schemaLocation="https://analysiscenter.veracode.com
/schema/4.0/dynamicscaninfo

https://analysiscenter.veracode.com/resource/
```

```

4.0&#x2f;dynamicscaninfo.xsd"
  account_id="<account id>" app_id="<app id>"
  scan_id="5097076" error_message="">
    <dynamic_scan scan_id="5097076"
      scan_name="Test DynamicDS Scan" scan_status="Incomplete"/>
  </dynamic_scan_info>

```

dynamicincludeexclude.do

The `dynamicincludeexclude.do` call configures a DynamicDS scan request to include and exclude specified URLs. Call the API once for each URL you want to include or exclude.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/5.0/dynamicincludeexclude.do>

Permissions

You need the [Upload and Scan API role](#) to use this call.

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		
url	String	URL you want to include in or exclude from the scan.
Required		
is_exclude	Boolean	If true, excludes the specified URL from the scan. If false, include. Default is false.
https_http_inclusion	Boolean	If true, applies the call settings to both the HTTP and HTTPS. If false, only applies the call settings to the supplied URL. Default is true.
directory_restriction_policy	String	Values include: <ul style="list-style-type: none"> <code>dir_only</code> Exclude a directory path. Ensure the last character of the <code>url</code> parameter value is a slash. <code>dir_and_sub</code> Exclude a directory path, including its subdirectories. Ensure the last character of the <code>url</code> parameter value is a slash. <code>no_restrict</code> Include the entire website. <code>file</code> Exclude a filepath. Ensure there is no slash after the last character of the <code>url</code> parameter value. Default is <code>dir_only</code> .

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/dynamicincludeexclude.do"
"app_id==<app id>" "url==http://www.example.com/archives/"
"is_exclude==true" "https_http_inclusion==true"
"directory_restriction_policy==dir_only"
```

HTTPIe Results

The `dynamicincludeexclude.do` call returns the `dynamic_scan_info` XML document, which references the [dynamicscaninfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `dynamicscaninfo.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<dynamic_scan_info xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
#x2f;4.0&#x2f;dynamicscaninfo"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;4.0&#x2f;dynamicscaninfo

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
4.0&#x2f;dynamicscaninfo.xsd"
  account_id="<account id>" app_id="<app id>"
  scan_id="5097076" error_message="">
    <dynamic_scan scan_id="5097076" scan_name="Test DynamicDS Scan"
  scan_status="Incomplete"
  target_url="http&#x3a;&#x2f;&#x2f;www.example.com&#x2f;"
  directory_restriction_policy="true"
  https_http_inclusion="true">
      <contact_information first_name="Joan" last_name="Smythe"
  telephone="123-456-7890" email="jsmythe&#x40;example.com"/>
      <allowed_hosts>
        <allowed_host
  host="http&#x3a;&#x2f;&#x2f;www.example.com&#x2f;"
  directory_restriction_policy="dir_and_sub"
  https_http_inclusion="true"/>
        </allowed_hosts>
      <exclude_urls>
        <exclude_url
  host="http&#x3a;&#x2f;&#x2f;www.example.com&#x2f;archives&#x2f;"
  directory_restriction_policy="dir_only"
  https_http_inclusion="true"/>
        </exclude_urls>
      </dynamic_scan>
    </dynamic_scan_info>
```

submitdynamicscan.do

The `submitdynamicscan.do` call submits a DynamicDS scan request.

Resource URL

`https://analysiscenter.veracode.com/api/5.0/submitdynamicscan.do`

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Permissions

You need the [Upload API and Upload API - Submit Only roles](#) to use this call.

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		
start_time	String	ISO 8601-compliant combined date and time. Optional, unless end_time is provided. Submitting an end_time without a start_time is submitting scan immediately.
end_time	String	ISO 8601-compliant combined date and time. Optional, unless start_time is provided. Submitting a start_time without an end_time is submitting scan immediately. Scan length default is one day if start_time and end_time are both provided.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/submitdynamicscan.do"
"app_id==<app id>" "start_time==2019-10-01T15:30:00-05:00"
"end_time==2019-10-01T16:30:00-05:00"
```

HTTPIe Results

The submitdynamicscan.do call returns the dynamic_scan_info XML document, which references the [dynamicscaninfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the dynamicscaninfo.xsd [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<dynamic_scan_info xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/4.0/dynamicscaninfo"

xsi:schemaLocation="https://analysiscenter.veracode.com/schema/4.0/dynamicscaninfo

https://analysiscenter.veracode.com/resource/4.0/dynamicscaninfo.xsd"
```

```

        account_id="<account id>" app_id="<app id>"
        scan_id="5097076" error_message="">
        <dynamic_scan scan_id="5097076" scan_name="Test DynamicDS Scan"
        scan_status="Submitted to Engine"
            target_url="http&#x3a;&#x2f;&#x2f;www.example.com&#x2f;"
            directory_restriction_policy="true"
            https_http_inclusion="true">
            <contact_information first_name="Joan" last_name="Smythe"
            telephone="123-456-7890" email="jsmythe&#x40;example.com"/>
            <allowed_hosts>
                <allowed_host host="http&#x3a;&#x2f;&#x2f;example.com&#x2f;"
            directory_restriction_policy="dir_and_sub"
                https_http_inclusion="true"/>
            </allowed_hosts>
            <exclude_urls>
                <exclude_url
            host="http&#x3a;&#x2f;&#x2f;example.com&#x2f;archives&#x2f;"
            directory_restriction_policy="dir_only"
                https_http_inclusion="true"/>
            </exclude_urls>
            <scan_schedule start_date="2019-10-01-04&#x3a;00"
            start_time="16&#x3a;30&#x3a;00-04&#x3a;00"
            end_date="2019-10-01-04&#x3a;00"
                end_time="17&#x3a;30&#x3a;00-04&#x3a;00"/>
            </dynamic_scan>
        </dynamic_scan_info>

```

addbrowserbasedlogin.do

The `addbrowserbasedlogin.do` call enables you to provide username and password credentials for browser-based logins when performing a DynamicDS scan.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/addbrowserbasedlogin.do`

Permissions

You need the [Upload API or Upload API - Submit Only role](#) to use this call.

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		
username	String	Username for logging in to the target website.
Required		
password	String	Password for logging in to the target website.
Required		

Name	Type	Description
windows_domain	String	Windows domain name, if required, for logging in to the target

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/addbrowserbasedlogin.do"
"app_id==<app id>" "username==<site username>" "password==<site
password>" "windows_domain==<site login domain>"
```

HTTPIe Results

The addbrowserbasedlogin.do call returns the dynamic_scan_info XML document, which references the [dynamicscaninfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the dynamicscaninfo.xsd [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<dynamic_scan_info xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/4.0/dynamicscaninfo"

xsi:schemaLocation="https://analysiscenter.veracode.com/schema/4.0/dynamicscaninfo

https://analysiscenter.veracode.com/resource/4.0/dynamicscaninfo.xsd"
  account_id="<account id>" app_id="<app id>"
  scan_id="5028209" error_message=""
    <dynamic_scan scan_id="5028209" scan_name="23 Sep 2019 Dynamic"
      scan_status="Incomplete"
        target_url="http://tester.co/"
        directory_restriction_policy="true" https_http_inclusion="false">
          <contact_information first_name="Joan" last_name="Smythe"
            telephone="123-456-7890" email="jsmythe@example.com"/>
          <allowed_hosts>
            <allowed_host host="http://example.com/"
              directory_restriction_policy="dir_and_sub"
              https_http_inclusion="true"/>
          </allowed_hosts>
          <login>
            <browser_based_login username="<site username>"
              password="<site password>" windows_domain="<site login domain>"/>
          </login>
        </dynamic_scan>
  </dynamic_scan_info>
```

uploadformbasedloginscript.do

The `uploadformbasedloginscript.do` call enables you to upload a forms-based login script for a DynamicDS scan.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/
uploadformbasedloginscript.do`

Permissions

You need the [Upload API or Upload API - Submit Only role](#) to use this call.

Parameters

Name	Type	Description
<code>app_id</code> Required	Integer	Application ID.
<code>login_script</code> Required	File	A login script captures the steps required to log in to the application. For more information about creating login scripts, see Providing Login Instructions .
<code>verification_url</code> Required	String	A URL Veracode can access after successfully logging in. See Providing Login Instructions for more information.
<code>verification_text</code> Required	String	A text string that Veracode finds at the specified verification URL after a successful login. See Providing Login Instructions for more information.

HTTPIE Example

Examples use the HTTPie command-line tool. See [Using HTTPie with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -f "https://  
analysiscenter.veracode.com/api/5.0/uploadformbasedloginscript.do"  
"app_id==<app id>" "login_script@siteloginscript.html"  
"verification_url=http://www.example.com/index.php"  
"verification_text=Welcome"
```

HTTPIE Results

The `uploadformbasedloginscript.do` call returns the `dynamic_scan_info` XML document, which references the [dynamicscaninfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `dynamicscaninfo.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<dynamic_scan_info xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/4.0/dynamicscaninfo"

xsi:schemaLocation="https://analysiscenter.veracode.com/schema/4.0/dynamicscaninfo

https://analysiscenter.veracode.com/resource/4.0/dynamicscaninfo.xsd"
  account_id="<account id>" app_id="<app id>"
  scan_id="5114591" error_message="">
    <dynamic_scan scan_id="5114591" scan_name="DynamicDS Scan with
Script" scan_status="Incomplete"
      target_url="http://dvwa.sa.veracode.io/"
      directory_restriction_policy="true"
      https_http_inclusion="true">
        <contact_information first_name="Joan" last_name="Smythe"
telephone="123-456-7890" email="jsmythe@example.com"/>
        <allowed_hosts>
          <allowed_host host="http://example.com/"
directory_restriction_policy="dir_and_sub"
            https_http_inclusion="true"/>
        </allowed_hosts>
        <exclude_urls>
          <exclude_url
host="http://www.example.com/archives/"
directory_restriction_policy="dir_only"
            https_http_inclusion="true"/>
        </exclude_urls>
        <login>
          <script_based_login login_sequence="siteloginscript.html"
verify_url="http://www.example.com/index.php"
            verify_string="Welcome"/>
        </login>
      </dynamic_scan>
</dynamic_scan_info>

```

getdynamicstatus.do

The `getdynamicstatus.do` call returns the status of the specified DynamicDS scan.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/getdynamicstatus.do`

Permissions

You need the [Upload and Scan API role](#) to use this call.

Parameters

Name	Type	Description
app_id	Integer	Application ID.

Name	Type	Description
Required		
build_id	Integer	Application or sandbox build ID.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getdynamicstatus.do"
"app_id=<app id>"
```

HTTPIe Results

The `getdynamicstatus.do` call returns the `scan_status_info` XML document, which references the [dynamicscanstatus.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `dynamicscanstatus.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<scan_status_info xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/4.0/dynamicscanstatus"

xsi:schemaLocation="https://analysiscenter.veracode.com
/schema/4.0/dynamicscanstatus

https://analysiscenter.veracode.com/resource/4.0/dynamicscanstatus.xsd"
  account_id="<account id>" app_id="<app id>"
  scan_id="5097076">
  <scan_status>Scan In Process</scan_status>
  <start>10/01/2019 4:46 PM EDT</start>
  <duration>00.0/26.0/21.0</duration>
  <state>In Process</state>
  <request>4000</request>
  <responses>4000</responses>
  <bytes_sent>1539.0 KB</bytes_sent>
  <bytes_received>1963.0 KB</bytes_received>
  <links>5</links>
  <login_failures>0</login_failures>
  <logins>0</logins>
  <network_errors>1</network_errors>
  <unreachable_hosts>0</unreachable_hosts>
</scan_status_info>
```

getdynamicflaws.do

The `getdynamicflaws.do` call returns information on a specific flaw.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/5.0/getdynamicflaws.do>

Permissions

You need the [Results API role](#) to use this call.

Parameters

Name	Type	Description
build_id	Integer	Application or sandbox build ID.
Required		
flaw_id	Integer	Find flaw IDs on the Triage Flaws page in the Veracode Platform. Fields returned by the Detailed Report API .
Required		

To locate the parameters for `getdynamicflaws.do`:

- 1 Call `getapplist.do` to locate the `app_id` for your application.
- 2 Call `getbuildlist.do` with the `app_id` to locate the current `build_id` for your application.
- 3 Call `detaileddetail.do` with the `build_id` to locate a `flaw_id`.
- 4 In the XML code of the detailed report, find the report section for the target flaw and then find the `issueid` element in that section. The `issueid` is the `flaw_id`.

HTTPIE Example

Examples use the HTTPie command-line tool. See [Using HTTPie with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getdynamicflaws.do"
"build_id=<build id>" "flaw_id=2"
```

HTTPIE Results

The `getdynamicflaws.do` call returns the `dynamicfinding` XML document, which references the [dynamicscaninfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `dynamicscaninfo.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<dynamicfinding xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/4.0/dynamicfinding"
```

```

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;4.0&#x2f;dynamicfinding

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
4.0&#x2f;dynamicscaninfo.xsd"
  app_id="<app id>" build_id="<build id>" flaw_id="2"
  engine_version=" " cwe_id="402"
    description="The security cookie does not have the
&#x22;HttpOnly&#x22; attribute set. Using this attribute helps
    to prevent client-side Javascript from accessing the
    cookie, thereby mitigating one of the most common XSS exploit
    scenarios.&#xa;" remediation="Unless the application
    requires that cookies be accessible to Javascript code, set
    the &#x22;HttpOnly&#x22; attribute when generating
    cookies.&#xa;"
    parameter_type=" "
    parameter_name=" "
    parameter="Set-Cookie security HTTP response header"
    original_arg=" "
    raw_response="HTTP&#x2f;1.1 302 Found&#xd;&#xa;Date&#x3a;
Tue, 01 Oct 2019 20&#x3a;46&#x3a;15 GMT&#xd;&#xa;Server&#x3a;
Apache&#xd;&#xa;Set-Cookie&#x3a; PHPSESSID&#x3d;
3696on897sncfp18jb800jnvfl&#x3b;
path&#x3d;&#x2f;&#xd;&#xa;Expires&#x3a;
Thu, 19 Nov 1981 08&#x3a;52&#x3a;00 GMT&#xd;&#xa;Cache-
Control&#x3a; no-store, no-cache, must-
revalidate&#xd;&#xa;Pragma&#x3a;
no-cache&#xd;&#xa;Set-Cookie&#x3a; PHPSESSID&#x3d;
3696on897sncfp18jb800jnvfl&#x3b; path&#x3d;&#x2f;&#xd;&#xa;Set-
Cookie&#x3a;
security&#x3d;low&#xd;&#xa;Location&#x3a;
login.php&#xd;&#xa;Keep-Alive&#x3a; timeout&#x3d;65, max&#x3d;
100&#xd;&#xa;Connection&#x3a;
Keep-Alive&#xd;&#xa;Content-Type&#x3a;
text&#x2f;html&#x3b; charset&#x3d;UTF-8&#xd;&#xa;Content-Length&#x3a;
0&#xd;&#xa;&#xd;&#xa;"
    injected_arg=" " referer_url=" ">
    <request host="www.example.com" port="80" secure="false"
raw_request="GET &#x2f; HTTP&#x2f;1.1&#xd;&#xa;Host&#x3a;
www.example.com&#xd;&#xa;User-Agent&#x3a; Mozilla&#x2f;
5.0 &#x28;Windows NT 5.2&#x3b; WOW64&#x3b; rv&#x3a;21.0&#x29;
Gecko&#x2f;20100101 Firefox&#x2f;21.0&#x2f;Veracode
Security Scan&#x2f;support&#x40;veracode.com&#xd;&#xa;Accept&#x3a;
&#x2a;&#x2f;&#x2a;&#xd;&#xa;Connection&#x3a; keep-
alive&#xd;&#xa;Accept-Encoding&#x3a; identity&#xd;&#xa;Accept-
Language&#x3a;
en-us,en&#x3b;q&#x3d;0.5&#xd;&#xa;Content-Length&#x3a;
0&#xd;&#xa;&#xd;&#xa;" method="GET" protocol="HTTP"
url="http&#x3a;&#x2f;&#x2f;dvwa.sa.veracode.io&#x2f;"
path="&#x2f;" uri="&#x2f;" body=" ">
    <header name="Host" value="www.example.com"/>
    <header name="User-Agent" value="Mozilla&#x2f;5.0 &#x28;Windows
NT 5.2&#x3b; WOW64&#x3b; rv&#x3a;21.0&#x29; Gecko&#x2f;20100101
Firefox&#x2f;21.0&#x2f;Veracode Security
Scan&#x2f;support&#x40;veracode.com"/>
    <header name="Accept" value="&#x2a;&#x2f;&#x2a;" />
    <header name="Connection" value="keep-alive"/>
    <header name="Accept-Encoding" value="identity"/>
    <header name="Accept-Language" value="en-us,en&#x3b;q&#x3d;
0.5"/>
    <header name="Content-Length" value="0"/>

```



```
</request>
</dynamicfinding>
```

getvsalist.do

The `getvsalist.do` call returns a list of available Virtual Scan Appliances (VSAs).

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/5.0/getvsalist.do>

Permissions

You need the [Upload and Scan API role](#) to use this call.

Parameters

This call takes no parameters.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getvsalist.do"
```

HTTPIe Results

The `getvsalist.do` call returns the `vsalist` XML document, which references the [vsalist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the [vsalist.xsd schema documentation](#).

The returned information includes the status of each available VSA. These statuses include:

- `notvsa`: the scan engine is not a VSA.
- `alpha`: the Veracode Platform cannot schedule jobs on this VSA.
- `uninitialized`: the VSA is not initialized.
- `operational`: the VSA is configured and ready to accept jobs.

```
<?xml version="1.0" encoding="UTF-8"?>

<vsalist xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;4.0&#x2f;vsalist"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;4.0&#x2f;vsalist
```

```
https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
4.0&#x2f;vsalist.xsd" account_id="31755">
  <appliance vsa_id="<vsa id>" vsg_id="<vsg id>"
vsa_status="uninitialized" vsa_status_desc="The VSA has not been
initialized."
  vsa_name="Vsa 1" vsa_desc="Test Desc 1"/>
  <appliance vsa_id="<vsa id>" vsg_id="<vsg id>"
vsa_status="operational" vsa_status_desc="The VSA is configured and
ready to accept jobs."
  vsa_name="Vsa 2" vsa_desc="Test Desc 2"/>
</vsalist>
```

assignvsa.do

The `assignvsa.do` call assigns a VSA to a DynamicDS scan request.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/5.0/assignvsa.do>

Permissions

You need the [Upload and Scan API role](#) to use this call.

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		
vsg_id	Integer	The VSA group ID.
Required		

HTTPIE Example

Examples use the HTTPie command-line tool. See [Using HTTPie with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/assignvsa.do" "app_id==<app id>"
"vsg_id==<vsg id>"
```

HTTPIe Results

The `assignvsa.do` call returns the `dynamic_scan_info` XML document, which references the [dynamicscaninfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `dynamicscaninfo.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<dynamic_scan_info xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="https://analysiscenter.veracode.com/schema/4.0/dynamicscaninfo"

xsi:schemaLocation="https://analysiscenter.veracode.com/schema/4.0/dynamicscaninfo

https://analysiscenter.veracode.com/resource/4.0/dynamicscaninfo.xsd"
  account_id="<account id>" app_id="<app id>"
  scan_id="5114591" error_message="">
    <dynamic_scan scan_id="5114591"
      scan_name="DynamicDS Scan with Script"
      scan_status="Incomplete"
      target_url="http://dvwa.sa.veracode.io/"
      directory_restriction_policy="true"
      https_http_inclusion="true" vsg_id="<vsg id>"
      <contact_information first_name="Joan" last_name="Smythe"
        telephone="123-456-7890" email="jsmythe@example.com"/>
      <allowed_hosts>
        <allowed_host host="http://example.com/"
          directory_restriction_policy="dir_and_sub"
          https_http_inclusion="true"/>
      </allowed_hosts>
      <exclude_urls>
        <exclude_url
          host="http://example.com/archives/"
          directory_restriction_policy="dir_only" https_http_inclusion="true"/>
        </exclude_urls>
      <login>
        <script_based_login login_sequence="sitelogin.html"
          verify_url="http://example.com/index.php"
          verify_string="Welcome"/>
      </login>
    </dynamic_scan>
  </dynamic_scan_info>
```

Flaw Report APIs

Using the Flaw Report API

The Flaw Report API has two calls that enable you to generate and download a summarized history of all findings for one, several, or all applications.

Veracode provides this API to simplify the process of identifying the current status, such as new, fixed, open, or reopened, of all the findings for a given application by returning one record per each finding discovered in the application history. The new calls are designed for integrating with external dashboards for vulnerability management.

The Flaw Report API comprises the following calls:

- [generateflawreport.do](#)** Returns `generateflawreport.xml`, which contains the token you need for downloading the finding report.
- [downloadflawreport.do](#)** Returns an XML report that lists all fixed and unfixed findings for the specified applications and/or scan type.

Prerequisites

Before using the Flaw Report API, you must meet the following prerequisites:

- A Veracode non-human [API user account](#) with the Archer API role. You cannot access the Archer API using a human account.
- [Veracode API ID and key credentials](#)

generateflawreport.do

The `generateflawreport.do` creates a report listing all fixed and unfixed flaws for the specified applications, scan types, or both. The return contains the token needed for downloading the flaw report.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/3.0/generateflawreport.do`

Permissions

You need the [Archer API role](#) to use this call.

Parameters

Name	Type	Description
<code>app_id_list</code> Required	Integer	Comma-separated list of the IDs for the applications you want included. This parameter does not support wildcards.
<code>scan_type</code>	String	Values include: <ul style="list-style-type: none"><code>static</code><code>dynamic</code><code>manual</code>

HTTPIE Example

Examples use the HTTPie command-line tool. See [Using HTTPie with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -o reporttoken.xml "https://analysiscenter.veracode.com/api/3.0/generateflawreport.do"
"app_id_list==<app1 id>,<app2 id>,<app3 id>" "scan_type==static"
```

HTTPIe Results

The `generateflawreport.do` call initiates the process of creating the `generateflawreport` XML document, which references the [archerreportrequest.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `archerreportrequest.xsd` [schema documentation](#).

The XML return contains the token string you need to retrieve the report, when it is available, using the `downloadflawreport.do` call.

```
<?xml version="1.0" encoding="UTF-8"?>

<archerreport xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;1.0&#x2f;archerapi"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;1.0&#x2f;archerapi

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
1.0&#x2f;archerreportrequest.xsd"
    token="4aaa2b4e-c42a-44c3-a696-c650a82d9c78"
    archer_report_version="3.0">
</archerreport>
```

downloadflawreport.do

The `downloadflawreport.do` call returns a generated XML report when it is available. This report lists all fixed and unfixed flaws for the specified applications, scan type, or both.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/3.0/downloadflawreport.do`

Permissions

You need the [Archer API role](#) to use this call.

Parameters

Name	Type	Description
token	Universally unique identifier (UUID)	Obtain this token from the XML report returned by calling generateflawreport.do . Use this token to download the generated flaw report. You can obtain the five most recent reports. Tokens expire after 30 days. If you do not provide a token, the call returns the latest report produced by generateflawreport.do , if one exists.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac -o flawreport.xml "https://
analysiscenter.veracode.com/api/3.0/downloadflawreport.do"
"token==7edf8d8f-f70d-4b73-97ad-f78db97f50f9"
```

HTTPIe Results

The downloadflawreport.do call returns the flaw report as an XML document, which references [archerreport.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the archerreport.xsd [schema documentation](#).

A partial sample of a flaw report.

```
<?xml version="1.0" encoding="UTF-8"?>

<Records xmlns="http://www.archer-tech.com/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:records_version="1.1"
  xsi:schemaLocation="https://analysiscenter.veracode.com/
    2.0/archerreport.xsd">
  <Record>
    <app_name>FlowMeter</app_name>
    <app_id>123456</app_id>
    <archer_app_name>1234</archer_app_name>
    <assurance_level>veryhigh</assurance_level>
    <teams>Demo Team</teams>
    <platform>Not Specified</platform>
    <version>13 Aug 2019 Static</version>
    <lifecycle_stage>Not Specified</lifecycle_stage>
    <rating>D</rating>
    <mitigated_rating></mitigated_rating>
    <static_score>82</static_score>
    <dynamic_score>0</dynamic_score>
    <manual_score>0</manual_score>
    <static_mitigated_score>0</static_mitigated_score>
    <dynamic_mitigated_score>0</dynamic_mitigated_score>
    <manual_mitigated_score>0</manual_mitigated_score>
    <app_origin>Not Specified</app_origin>
    <generation_date>2019-08-13-14</generation_date>
    <planned_deployment_date></planned_deployment_date>
    <last_update_date>2019-08-13-14</last_update_date>
    <submitted_date>2019-08-13-13</submitted_date>
    <policy_name>Veracode Transitional Very High</policy_name>
    <policy_version>1</policy_version>
    <policy_compliance_status>Did Not Pass</
  policy_compliance_status>
    <policy_rules_passed>false</policy_rules_passed>
    <grace_period_expired>true</grace_period_expired>
    <scan_overdue>true</scan_overdue>
    <business_owner></business_owner>
```

```

<business_unit>Not Specified</business_unit>
<tags></tags>
<custom0></custom0>
<custom1></custom1>
<custom2></custom2>
<custom3></custom3>
<custom4></custom4>
<customfield name="Custom 1" value=""/>
<customfield name="Custom 2" value=""/>
<customfield name="Custom 3" value=""/>
<customfield name="Custom 4" value=""/>
<customfield name="Custom 5" value=""/>
<customfield name="Custom 6" value=""/>
<customfield name="Custom 7" value=""/>
<customfield name="Custom 8" value=""/>
<customfield name="Custom 9" value=""/>
<customfield name="Custom 10" value=""/>
<any_scan_due_date></any_scan_due_date>
<flaws>
  <Record>
    <app_name>Apache</app_name>
    <app_id>575960</app_id>
    <archer_app_name>2502</archer_app_name>
    <version>13 Aug 2019 Static</version>
    <platform>Not Specified</platform>
    <flaw_issue_id>1</flaw_issue_id>
    <module>httpd</module>
    <severity>2</severity>
    <type>Unchecked Error Condition</type>
    <flaw_description>&#x3c;span&#x3e;The result of this call
to calloc&#x28;&#x29; is not checked for
    success before being used. This can result in
application instability or crashing if memory is
    not available.&#x3c;&#x2f;span&#x3e;
&#x3c;span&#x3e;Be sure to check the result and ensure it
    is correct before use. Some functions return a pointer
which should be validated as not NULL before
    use. Other functions return integers or Boolean
values that must either be zero or non-zero for the
    results of the function to be used. Consult the API
documentation to determine what a correct result
    is from the function call.&#x3c;&#x2f;span&#x3e;
&#x3c;span&#x3e;References&#x3a; &#x3c;a
href&#x3d;&#x22;https&#x3a;&#x2f;&#x2f;cwe.mitre.org&#x2f;data&#x2f;de
finitions&#x2f;391.html&#x22;&#x3e;
    CWE&#x3c;&#x2f;a&#x3e; &#x3c;a
href&#x3d;&#x22;https&#x3a;&#x2f;&#x2f;www.owasp.org&#x2f;index.php&#x
2f;
Improper_Error_Handling&#x22;&#x3e;OWASP&#x3c;&#x2f;a&#x3e;&#x3c;&#x2f;
span&#x3e;</flaw_description>
    <note></note>
    <cweid>391</cweid>
    <remediationeffort>2</remediationeffort>
    <exploitLevel>-1</exploitLevel>
    <sourcefile>scoreboard.c</sourcefile>
    <line>120</line>

    <sourcefilepath>home&#x2f;efeller&#x2f;httpd-2.2.8&#x2f;server&#x2f;sc
oreboard.c</sourcefilepath>

```

```

        <scope>UNKNOWN</scope>
        <functionprototype>void ap_init_scoreboard&#x28;void
&#x2a;&#x29;</functionprototype>
        <functionrelativelocation>15</functionrelativelocation>
        <url></url>
        <categoryid>16</categoryid>
        <categoryname>Error Handling</categoryname>
        <pcirelated>false</pcirelated>
        <cwe_description>&#x5b;PLANNED FOR DEPRECATION. SEE
MAINTENANCE NOTES.&#x5d; Ignoring exceptions and other error
conditions may allow an attacker to induce unexpected behavior
unnoticed.</cwe_description>
        <count>1</count>
        <capecid>0</capecid>
        <exploitdifficulty>0</exploitdifficulty>
        <exploit_desc></exploit_desc>
        <severity_desc></severity_desc>
        <remediation_desc></remediation_desc>
        <date_first_occurrence>2019-08-13-13&#x3a;55</
date_first_occurrence>
        <date_first_occurrence>2019-08-13-13&#x3a;55</
date_first_occurrence>
        <remediation_status>New</remediation_status>
        <cia_impact>nnp</cia_impact>
        <is_latest_build>true</is_latest_build>
        <published_date>2019-08-13-14&#x3a;08</published_date>
        <affects_policy_compliance>false</
affects_policy_compliance>
        <mitigation_status>none</mitigation_status>
        <mitigation_status_desc>Not Mitigated</
mitigation_status_desc>
        <mitigations/>
    </Record>
    ...

```

Sandbox APIs

Using the Sandbox APIs

You can use the Sandbox API calls to create, list, promote, update, and delete developer sandboxes.

- [createsandbox.do](#)** The `createsandbox.do` call creates a sandbox for the specified application.
- [getsandboxlist.do](#)** The `getsandboxlist.do` call returns a list of all the sandboxes associated with the specified application.
- [promotesandbox.do](#)** The `promotesandbox.do` call promotes the specified build ID to be the policy scan for the application, as long as that build has had a successful scan.
- [updatesandbox.do](#)** The `updatesandbox.do` call enables you to amend an existing sandbox in the application profile.
- [deletesandbox.do](#)** The `deletesandbox.do` call deletes a sandbox from the specified application.

createsandbox.do

The `createsandbox.do` call creates a sandbox for the specified application. If you have already reached the maximum number of sandboxes for the application, this call fails.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/createsandbox.do`

Parameters

Name	Type	Description
<code>app_id</code> Required	Integer	Application ID.
<code>sandbox_name</code> Required	String	The unique identifier for the new sandbox.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/createsandbox.do" "app_id=<app
id>" "sandbox_name==Project Security"
```

HTTPIe Results

The `createsandbox.do` call returns the `sandboxinfo` XML document, which references the [sandboxinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `sandboxinfo.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<sandboxinfo xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;4.0&#x2f;sandboxinfo"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;4.0&#x2f;sandboxinfo

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
4.0&#x2f;sandboxinfo.xsd" sandboxinfo_version="1.2"
  account_id="<account id>" app_id="<app id>"
  <sandbox sandbox_id="<sandbox id>" sandbox_name="Project Security"
  sandbox_status="sandbox" owner="<Veracodeusername>"
    modified_date="2019-09-17T14&#x3a;08&#x3a;35-04&#x3a;00"
```

```

created_date="2019-09-17T14&#x3a;08&#x3a;35-04&#x3a;00">
  <customfield name="Custom 1" value="" />
  <customfield name="Custom 2" value="" />
  <customfield name="Custom 3" value="" />
  <customfield name="Custom 4" value="" />
  <customfield name="Custom 5" value="" />
</sandbox>
</sandboxinfo>

```

If you exceed the maximum number of sandboxes for the application, the return contains:

```

<?xml version="1.0" encoding="UTF-8"?>
<error>An application can have up to 25 sandboxes. To create a new
sandbox, you must delete one.</error>

```

getsandboxlist.do

The `getsandboxlist.do` call returns the list of sandboxes associated with the specified application.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/5.0/getsandboxlist.do>

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```

http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/getsandboxlist.do" "app_id==<app
id>"

```

HTTPIe Results

The `getsandboxlist.do` call returns the `sandboxlist` XML document, which references the [sandboxlist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `sandboxlist.xsd` [schema documentation](#).

```

<?xml version="1.0" encoding="UTF-8"?>

<sandboxlist xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

```

```

xmlns="https://analysiscenter.veracode.com/schema/4.0/sandboxlist"

xsi:schemaLocation="https://analysiscenter.veracode.com/schema/4.0/sandboxlist

https://analysiscenter.veracode.com/resource/4.0/sandboxlist.xsd"
  sandboxlist_version="1.0" account_id="<account id>"
app_id="<app id>"
  <sandbox sandbox_id="< sandbox id>" sandbox_name="Project
Security" owner="<Veracodeusername>"
    last_modified="2019-09-17T14:08:35-04:00">
    <customfield name="Custom 1" value=""/>
    <customfield name="Custom 2" value=""/>
    <customfield name="Custom 3" value=""/>
    <customfield name="Custom 4" value=""/>
    <customfield name="Custom 5" value=""/>
  </sandbox>
  <sandbox sandbox_id="< sandbox id>" sandbox_name="Project Refactor"
owner="<Veracodeusername>"
    last_modified="2019-09-17T14:04:13-04:00">
    <customfield name="Custom 1" value=""/>
    <customfield name="Custom 2" value=""/>
    <customfield name="Custom 3" value=""/>
    <customfield name="Custom 4" value=""/>
    <customfield name="Custom 5" value=""/>
  </sandbox>
</sandboxlist>

```

promotesandbox.do

If the specified sandbox build scans successfully, the `promotesandbox.do` call promotes the build to be the policy scan for the application.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/promotesandbox.do`

Parameters

Name	Type	Description
build_id	Integer	ID of the latest build for the target sandbox.
Required		

You can identify your `build_id` by calling [getbuildlist.do](#).

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/promotesandbox.do"
"build_id=<build id>"
```

HTTPIe Results

The `promotesandbox.do` call returns the `sandboxinfo` XML document, which references the [sandboxinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `sandboxinfo.xsd` [schema documentation](#).

In the returned XML, the `sandbox_status="policy"` attribute indicates that the API successfully promoted the sandbox build.

```
<?xml version="1.0" encoding="UTF-8"?>

<sandboxinfo xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
#x2f;4.0&#x2f;sandboxinfo"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;4.0&#x2f;sandboxinfo

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
4.0&#x2f;sandboxinfo.xsd" sandboxinfo_version="1.2"
  account_id="<account id>" app_id="<app id>" build_id="<build
id>" analysis_id="4978995" analysis_unit_id="4994637">
    <sandbox sandbox_id="<sandbox id>" sandbox_status="policy"
owner="<Veracodeusername>"
      modified_date="2019-09-18T17&#x3a;11&#x3a;45-04&#x3a;00"
created_date="2019-09-18T17&#x3a;11&#x3a;45-04&#x3a;00">
      <customfield name="Custom 1" value=""/>
      <customfield name="Custom 2" value=""/>
      <customfield name="Custom 3" value=""/>
      <customfield name="Custom 4" value=""/>
      <customfield name="Custom 5" value=""/>
    </sandbox>
  </sandboxinfo>
```

updatesandbox.do

The `updatesandbox.do` call allows you to update the custom field values for the specified sandbox.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Permissions

You must have the [Creator](#), [Security Lead](#), or [Sandbox Administrator API](#) role to use this API.

Resource URL

`https://analysiscenter.veracode.com/api/5.0/updatesandbox.do`

Parameters

Name	Type	Description
<code>sandbox_id</code> Required	Integer	ID of the target sandbox.
<code>custom_field_name</code> Required	String	Specifies the custom field to update with the <code>custom_field_value</code> . You can change one custom field value per call to <code>updatesandbox.do</code> .
<code>custom_field_value</code> Required	String	Specifies the <code>custom_field_value</code> to store in the specified custom field. You can change one custom field value per call to <code>updatesandbox.do</code> .

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/updatesandbox.do"
"sandbox_id==<sandbox id>" "custom_field_name==Custom 2"
"custom_field_value==Work on hold."
```

HTTPIe Results

The `updatesandbox.do` call returns the `sandboxinfo` XML document, which references the [sandboxinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `sandboxinfo.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<sandboxinfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="https://analysiscenter.veracode.com/schema/4.0/sandboxinfo"
  xsi:schemaLocation="https://analysiscenter.veracode.com/schema/4.0/sandboxinfo
https://analysiscenter.veracode.com/resource/4.0/sandboxinfo.xsd"
  sandboxinfo_version="1.2" account_id="<account id>"
  app_id="<app id>"
  <sandbox sandbox_id="<sandbox id>" sandbox_name="Project Security"
  sandbox_status="sandbox" owner="<Veracodeusername>"
    modified_date="2019-09-17T14:08:35-04:00"
    created_date="2019-09-17T14:08:35-04:00">
    <customfield name="Custom 1" value=""/>
    <customfield name="Custom 2" value="Work on hold."/>
```

```

    <customfield name="Custom 3" value="" />
    <customfield name="Custom 4" value="" />
    <customfield name="Custom 5" value="" />
  </sandbox>
</sandboxinfo>

```

deletesandbox.do

The `deletesandbox.do` call deletes the specified sandbox.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/5.0/deletesandbox.do`

The `deletesandbox.do` call deletes a sandbox and its contents. You cannot undo the deletion.

Permissions

Non-human API user accounts require the [Upload and Scan role](#) to use this call. Human user accounts require the [Creator, Security Lead, or Sandbox Administrator role](#) to use this call. If you have delete permission, you do not have to own the sandbox to delete it. If you are a team-restricted user, you can only delete sandboxes owned by your teams.

Parameters

Name	Type	Description
sandbox_id	Long	ID of the target sandbox.
Required		

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```

http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/5.0/deletesandbox.do"
"sandbox_id=1554802"

```

HTTPIe Results

The `deletesandbox.do` call returns the `sandboxlist` XML document, which references the [sandboxlist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `sandboxlist.xsd` [schema documentation](#).

In the returned list of sandboxes for the application that owned the deleted sandbox, the deleted sandbox does not appear.

```

<?xml version="1.0" encoding="UTF-8"?>

```

```
<sandboxlist xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;4.0&#x2f;sandboxlist"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;4.0&#x2f;sandboxlist

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
4.0&#x2f;sandboxlist.xsd"
    sandboxlist_version="1.0" account_id="<account id>"
app_id="<app id>">
  <sandbox sandbox_id="1507884" sandbox_name="Csharp1sb"
owner="jsmythe@example.com" last_modified="2019-10-18T12&#x3a;38&#x3a;
19-04&#x3a;00">
    <customfield name="Custom 1" value="" />
    <customfield name="Custom 2" value="" />
    <customfield name="Custom 3" value="" />
    <customfield name="Custom 4" value="" />
    <customfield name="Custom 5" value="" />
  </sandbox>
</sandboxlist>
```

VAST APIs

Using the VAST APIs

The Veracode Vendor Application Security Testing (VAST) program has APIs for automating vendor and enterprise tasks.

These APIs include:

- | | |
|---|---|
| sharedreport.do | The <code>sharedreport.do</code> call returns the details of a shared report. |
| sharedreportpdf.do | The <code>sharedreportpdf.do</code> call returns the shared report as a PDF document. |
| getsharedreportlist.do | The <code>getsharedreportlist.do</code> call returns a list of published, shared reports that are linked to an application. |
| getsharedreportinfo.do | The <code>getsharedreportinfo.do</code> call returns information about a specific shared report. |

sharedreport.do

The `sharedreport.do` call returns the details of a shared report.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/4.0/sharedreport.do>

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		
shared_report_id	Integer	You can get the shared report ID from the sharedreportlist.do call.
Required		

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/4.0/sharedreport.do" "app_id=<app
id>" "shared_report_id=10651"
```

HTTPIe Results

The `sharedreport.do` call returns the `summaryreport` XML document, which references the [summaryreport.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `summaryreport.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<summaryreport xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;www.veracode.com&#x2f;schema&#x2f;report
s&#x2f;export&#x2f;1.0"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;www.veracode.com&#x2f;schem
a&#x2f;reports&#x2f;export&#x2f;1.0

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
summaryreport.xsd"
  report_format_version="1.3" account_id="<account id>"
  app_name="<app name>" app_id="<app id>"
  analysis_id="674675" static_analysis_unit_id="690423"
  sandbox_id="<sandbox id>"
  first_build_submitted_date="2015-12-22 15&#x3a;47&#x3a;59
UTC" version="22 July 2019 Static"
  build_id="<build id>" submitter="<vendor name>"
  platform="Not Specified" assurance_level="3"
  business_criticality="3" generation_date="2019-10-03 18&#x3a;
03&#x3a;20 UTC" veracode_level="VL1"
  total_flaws="366" flaws_not_mitigated="365" teams=""
  life_cycle_stage="Not Specified"
  planned_deployment_date="2019-12-22 15&#x3a;27&#x3a;13 UTC"
  last_update_time="2019-07-22 16&#x3a;05&#x3a;52 UTC"
  is_latest_build="true" policy_name="3rd Party"
  policy_version="2" policy_compliance_status="Did Not Pass"
  policy_rules_status="Did Not Pass"
```



```

    grace_period_expired="false" scan_overdue="false" business_owner=""
    business_unit="Not Specified" tags=""
    legacy_scan_engine="false">
    <static-analysis rating="C" score="53" submitted_date="2015-12-22
15&#x3a;47&#x3a;47 UTC"
    published_date="2015-12-22 16&#x3a;05&#x3a;49 UTC"
    version="22 July 2019 Static" mitigated_rating="C"
    mitigated_score="53" next_scan_due="2019-12-01 15&#x3a;
05&#x3a;49 UTC" analysis_size_bytes="5696667"
    engine_version="88693">
    <modules>
    <module name="<module name>" compiler="JAVAC_7" os="Java
J2SE 7" architecture="JVM" loc="77543" score="52"
    numflawssev0="1" numflawssev1="0" numflawssev2="22"
    numflawssev3="321" numflawssev4="20" numflawssev5="2"/>
    </modules>
    </static-analysis>
    <severity level="5">
    <category categoryname="Command or Argument Injection"
severity="Very High" count="2"/>
    </severity>
    <severity level="4">
    <category categoryname="SQL Injection" severity="High"
count="20"/>
    </severity>
    <severity level="3">
    <category categoryname="Cross-Site Scripting &#x28;XSS&#x29;"
severity="Medium" count="252"/>
    <category categoryname="Credentials Management"
severity="Medium" count="44"/>
    <category categoryname="CRLF Injection" severity="Medium"
count="8"/>
    <category categoryname="Cryptographic Issues" severity="Medium"
count="5"/>
    <category categoryname="Insufficient Input Validation"
severity="Medium" count="4"/>
    <category categoryname="Code Quality" severity="Medium"
count="3"/>
    <category categoryname="Directory Traversal" severity="Medium"
count="3"/>
    <category categoryname="Encapsulation" severity="Medium"
count="2"/>
    </severity>
    <severity level="2">
    <category categoryname="Information Leakage" severity="Low"
count="10"/>
    <category categoryname="Cryptographic Issues" severity="Low"
count="5"/>
    <category categoryname="Code Quality" severity="Low" count="5"/>
    <category categoryname="API Abuse" severity="Low" count="2"/>
    </severity>
    <severity level="1"/>
    <severity level="0">
    <category categoryname="Potential Backdoor"
severity="Informational" count="1"/>
    </severity>
    <flaw-status new="366" reopen="0" open="0" cannot-reproduce="0"
fixed="0" total="366" not_mitigated="365"
    sev-1-change="0" sev-2-change="22" sev-3-change="321" sev-4-
change="20" sev-5-change="2"/>
    <customfields>

```

```

<customfield name="JIRA" value=""/>
<customfield name="JenkinsID" value=""/>
<customfield name="SDLC Stage" value=""/>
<customfield name="Custom 4" value=""/>
<customfield name="Custom 5" value=""/>
<customfield name="Custom 6" value=""/>
<customfield name="Custom 7" value=""/>
<customfield name="Custom 8" value=""/>
<customfield name="Custom 9" value=""/>
<customfield name="Custom 10" value=""/>
</customfields>
<software_composition_analysis third_party_components="0"
violate_policy="false" components_violated_policy="0"
sca_service_available="false">
  <vulnerable_components/>
</software_composition_analysis>
</summaryreport>

```

sharedreportpdf.do

The `sharedreportpdf.do` call returns the shared report as a PDF document to the specified directory.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/4.0/sharedreportpdf.do`

Parameters

Name	Type	Description
<code>app_id</code> Required	Integer	Application ID.
<code>shared_report_id</code> Required	Integer	You can get the shared report ID from the <code>sharedreportlist.do</code> call.

HTTPIe Example

Examples use the HTTPIe command-line tool. See [Using HTTPIe with the Python Authentication Library](#) on page 16.

```

http --auth-type=veracode_hmac -o c:\veracodeapi\sharedreport.pdf
"https://analysiscenter.veracode.com/api/4.0/sharedreportpdf.do"
"app_id==<app id>" "shared_report_id==10651"

```

HTTPIe Results

The `sharedreportpdf.do` call returns the `summaryreport` PDF document.

getsharedreportlist.do

The `getsharedreportlist.do` call returns a list of published shared reports that are linked to the application.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

`https://analysiscenter.veracode.com/api/4.0/getsharedreportlist.do`

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		

HTTPIE Example

Examples use the HTTPie command-line tool. See [Using HTTPie with the Python Authentication Library](#) on page 16.

```
http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/4.0/getsharedreportlist.do"
"app_id=<app id>"
```

HTTPIE Results

Examples use the HTTPie command-line tool. See [Using HTTPie with the Python Authentication Library](#) on page 16.

The `getsharedreportlist.do` call returns the `sharedreportlist` XML document, which references the [sharedreportlist.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `sharedreportlist.xsd` [schema documentation](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<sharedreportlist xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
#x2f;3.0&#x2f;sharedreportlist"

xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;3.0&#x2f;sharedreportlist

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
3.0&#x2f;sharedreportlist.xsd"
  account_id="<account id>" app_id="<app id>" app_name="<app
name>" vendor_name="<vendor name>">
  <sharedreport shared_report_id="13391" report_name="Vendor A's
static scan"
    shared_date="2019-07-22T09&#x3a;06&#x3a;19-05&#x3a;00"/>
```

```

    <sharedreport shared_report_id="13288" report_name="Vendor A's
static scan"
    shared_date="2019-05-02T22&#x3a;20&#x3a;41-05&#x3a;00"/>
    <sharedreport shared_report_id="13007" report_name="Vendor A's
static scan"
    shared_date="2019-04-18T18&#x3a;40&#x3a;57-05&#x3a;00"/>

</sharedreportlist>

```

getsharedreportinfo.do

The `getsharedreportinfo.do` call returns information about a specific shared report. If you do not specify a shared report ID, it returns the latest shared report.

Before using this API, Veracode strongly recommends that you read [API Usage and Access Guidelines](#).

Resource URL

<https://analysiscenter.veracode.com/api/4.0/getsharedreportinfo.do>

Parameters

Name	Type	Description
app_id	Integer	Application ID.
Required		
shared_report_id	Integer	You can get the shared report ID from the <code>sharedreportlist.do</code> call.
Required		

HTTPIE Example

Examples use the HTTPie command-line tool. See [Using HTTPie with the Python Authentication Library](#) on page 16.

```

http --auth-type=veracode_hmac "https://
analysiscenter.veracode.com/api/4.0/getsharedreportinfo.do"
"app_id=<app id>" "shared_report_id=10651"

```

HTTPIE Results

The `getsharedreportinfo.do` call returns the `sharedreportinfo` XML document, which references the [sharedreportinfo.xsd](#) schema file. You can use the XSD schema file to validate the XML data. See the `sharedreportinfo.xsd` [schema documentation](#).

```

<?xml version="1.0" encoding="UTF-8"?>

<sharedreportinfo xmlns:xsi="http&#x3a;&#x2f;&#x2f;www.w3.org&#x2f;
2001&#x2f;XMLSchema-instance"

xmlns="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;schema&
&#x2f;3.0&#x2f;sharedreportinfo"

```

```
xsi:schemaLocation="https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com
&#x2f;schema&#x2f;3.0&#x2f;sharedreportinfo

https&#x3a;&#x2f;&#x2f;analysiscenter.veracode.com&#x2f;resource&#x2f;
3.0&#x2f;sharedreportinfo.xsd"
    account_id="<account id>" app_id="<app id>"
shared_report_id="10651">
    <sharedreport shared_report_id="10651" report_name="Vendor A's
static scan" policy_name="3rd Party" policy_version="2"
    policy_compliance_status="Did Not Pass" rule_status="Did Not
Pass" scan_status="Pass"
    shared_date="2019-07-22T09&#x3a;06&#x3a;19-05&#x3a;00">
    <analysis_unit analysis_type="Static" analysis_name="22 July
2019 Static"/>
    </sharedreport>
</sharedreportinfo>
```

API Wrappers

Using API Wrappers

Veracode provides API wrappers for Java and C# to simplify the integration of the Veracode APIs.

API wrappers are language-specific kits or packages that wrap sets of API calls into easy-to-use functions. The wrapper calls multiple API calls without the interaction of the user, further automating projects. Information about the individual [API calls](#) is also available and how to create an [API user](#) is available in the Veracode Help Center.

The Veracode API wrappers are Veracode-developed CLI programs that can communicate with the Veracode APIs accelerating the integration of the Veracode APIs in your software development lifecycle. The CLI programs are available in both C# and Java versions. The wrapper manages the details of network connections, parameters, and interfaces so that you can focus on using the objects to integrate Veracode into your code. The content of the wrappers is commonly used as library or command-line tools.

Veracode provides API wrappers for the following languages:

- [C#](#)
- [Java](#)



NOTE: If you are using the Veracode integrations to automate static analysis of your applications, your uploaded modules must not have fatal or blocking errors. These errors prevent Veracode Static Analysis from starting and cause your automation to fail. Before running your automation, [perform a prescan verification](#) to identify and resolve any errors in your modules and files.

Install and Use the C# API Wrapper

Before you can use the C# API wrapper, you must meet the software requirements listed in the [Veracode Integrations Support Matrix](#).

For instructions on referencing the C# API wrapper from your Visual Studio solution, see [Referencing the Veracode C# API Wrapper from Visual Studio](#) on page 202.

To use the C# API wrapper from the command line:

- 1 Download and extract the [C# ZIP file](#).



NOTE: The ZIP file contains a `Help` directory that contains documentation files. To access the documentation, open `index.html` from the `Help` directory.

- 2 In a command prompt window, navigate to the directory where the EXE file is located and enter `VeracodeC#API`.
- 3 Enter `VeracodeC#API -vid Veracode API ID -vkey Veracode API key -action action`, where `action` is any of the `-action` values listed under Usage in the Help window of the console.
- 4 Review the parsing errors to see if any parameters are missing or have invalid arguments.
- 5 To enter any missing parameters, enter `VeracodeC#API -vid Veracode API ID -vkey Veracode API key -action action-param1 arg1 -param2 arg2...`, where `param#` is a missing parameter and `arg#` is the corresponding argument.

You can use the optional parameter `-inputfilepath` if you want to provide the filepath of a CSV file from which to read additional command-line arguments. The correct format of the CSV file is to enter the parameter names in the first row and the corresponding values of those parameters in the subsequent rows.

You can use `inputfilepath` to make multiple calls to the API specified by the action. In the CSV file, use one row to specify the parameters for each call. For an example, see [createuser.do](#).

The wrapper returns a zero (integer) exit code when a command succeeds.

The wrapper returns a non-zero (integer) exit code when a command fails, as follows:

- 1 = Invalid input
- 2 = API internal error
- 3 = Incorrect file format of the CSV file referred to in the `-inputfilepath` parameter
- 4 = The scan did not pass [policy compliance](#). This code only applies to an `uploadandscan` composite action that specifies the `scantimeout` parameter.

The Veracode API wrappers return errors for missing required parameters and unrecognized parameters. The API wrappers do not return errors on defined API parameters that are not valid for use with the specified action. For example, if an API wrapper takes `sandboxid` as an optional parameter and you supply `sandboxname` in error, the API wrapper ignores `sandboxname` and executes. You can verify the list of valid parameters in the console or the XML API documentation.

Install and Use the Java API Wrapper

Before you can use the Java API wrapper, you must meet the software requirements listed in the [Veracode Integrations Support Matrix](#).

The Veracode Java API wrapper is available in Maven Central for you to add as a dependency in the build scripts of your projects. For example, you can add it to a Gradle or Maven project. You can also [reference the wrapper from your Eclipse project](#).

To use the Java API wrapper:

- 1 Go to <https://search.maven.org/search?q=a:vosp-api-wrappers-java>
The most recent build of the wrapper displays in the Search Results table.
- 2 Click **dist.zip** in the Download column to download `vosp-api-wrapper-java-<version>-dist.zip`.

If you want to download an older version of the wrapper, click **all** in the Latest Version column, then click **dist.zip** for the appropriate version.

- 3 Extract the ZIP file.

The ZIP file contains Javadoc files. In the `Help` directory, open `help-doc.html`.



- 4 In a command prompt window, enter `java -jar VeracodeJavaAPI.jar`.

Each artifact is associated with the MD5 and SHA1 checksums. You can use these checksum files to verify the integrity of the associated artifacts. You can access the console help from the Help window in the console window.

Verify the Authenticity of Java Artifacts

Each Java artifact is associated with an ASC signature file for verifying that the publication source is Veracode. You can download the ASC file for the appropriate version of the Java wrapper [here](#).

You have installed a GNU Privacy Guard (GPG) utility, such as GnuPG.

To verify that the publication source of Java artifacts is Veracode:

- 1 Download the Veracode public key from a public keyserver (such as pgp.mit.edu) using the key ID 0x63003CB3. For example:

```
gpg --keyserver pgp.mit.edu --recv-key 0x63003CB3
```

- 2 Verify the signature of an artifact. The following example is verifying the signature of the vosp-api-wrapper-java-17.10.4.8.jar (assuming it is in the same directory as the ASC file):

```
gpg --verify vosp-api-wrapper-java-17.10.4.8.jar.asc
```

The following output tells you that the Veracode public key is not trusted locally:

```
gpg: Signature made 11/02/17 14:49:01 Eastern Daylight Time
gpg:          using RSA key
E1AE087F8B51E8F322513009A0D8098560410C91
gpg: Good signature from "Veracode" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to
the owner.
Primary key fingerprint: 130D 4190 4800 95BD 01F5  F130 235A 4AC4
6300 3CB3
Subkey fingerprint: E1AE 087F 8B51 E8F3 2251  3009 A0D8 0985
6041 0C91
```

You can trust the Veracode public key and verify the signature of more artifacts, but Veracode recommends that you always compare the fingerprints from the output to the following fingerprints to ensure the signature is not forged.

```
pub  rsa2048 2017-11-02 [expires: 2020-11-01]
     130D 4190 4800 95BD 01F5  F130 235A 4AC4 6300 3CB3
uid  Veracode
sub  rsa2048 2017-11-02 [expires: 2020-11-01]
     E1AE 087F 8B51 E8F3 2251  3009 A0D8 0985 6041 0C91
```

Configure the Java Wrapper Parameters

To enter required parameters to run the API wrapper functions:

- 1 Enter `java -jar VeracodeJavaAPI.jar -vid Veracode API ID -vkey Veracode API key -action action`. You can list valid actions by running the API wrapper with the `-help` option.

```
java -jar VeracodeJavaAPI.jar -help
```

- 2 Review the parsing errors window to see if any parameters are missing or have invalid arguments.
- 3 To enter any missing parameters, enter `java -jar VeracodeJavaAPI.jar -vid Veracode API ID -vkey Veracode API key -action action -param1 arg1 -param2 arg2...`, where `param#` is a missing parameter and `arg#` is the corresponding argument.

In addition, you can use the optional parameter `-inputfilepath` if you want to provide the filepath of a CSV file from which to read additional command-line arguments. The correct format of the CSV file is to enter the parameter names in the first row and the corresponding values of those parameters in the subsequent rows.

You can use `inputfilepath` to make multiple calls to the API specified by the action. In the CSV file, use one row to specify the parameters for each call. For an example, see [createuser.do](#).

The wrapper returns a zero (integer) exit code when a command succeeds.

The wrapper returns a non-zero (integer) exit code when a command fails, as follows:

- 1 = Invalid input
- 2 = API internal error
- 3 = Incorrect file format of the CSV file referred to in the `-inputfilepath` parameter
- 4 = The scan did not pass [policy compliance](#). This code only applies to an `uploadandscan` composite action that specifies the `scantimeout` parameter.

The Veracode API wrappers return errors for missing required parameters and unrecognized parameters. The API wrappers do not return errors on defined API parameters that are not valid for use with the specified action. For example, if an API wrapper takes `sandboxid` as an optional parameter and you supply `sandboxname` in error, the API wrapper ignores `sandboxname` and executes. You can verify the list of valid parameters in the console or the XML API documentation.



NOTE: You can also reference the Java API wrapper from Eclipse. See [Referencing the Veracode Java API Wrapper from Eclipse](#) on page 202.

Using the Veracode API Wrappers from the Command Line

The Veracode APIs allow users to access the Veracode Platform and perform actions such as create application profiles, upload files, begin scans, and download reports. Users can interact with the Veracode APIs programmatically or through a web browser.

Like the Veracode APIs, the API wrappers allow users to access the Veracode Platform and perform different actions in a more simplified way that requires neither a web browser nor repetitive programming. When used as a library, the Veracode API wrappers enable programmers to no longer have to programmatically configure web requests and look up the URL and query string parameters of the Veracode API functions they want to call. When used as a command-line

application, the Veracode API wrappers allow users to perform single and composite actions that require no programming knowledge.

The Veracode API wrappers, when used from the command line, allow users to perform two kinds of actions: [simple](#) and [composite](#). Simple actions correspond to a single call to one of the functions of the Veracode API. Composite actions correspond to calls to multiple functions of the Veracode API.

Run the API Wrapper from the Command Line

To run the Veracode Java or C# API wrapper from the command line:

- 1 Open a command prompt.
- 2 Navigate to the directory where you [downloaded the wrapper](#).
- 3 Enter one of the following commands to run the Java or C# API wrapper:

- For Java: `java -jar VeracodeJavaAPI.jar`
- For C#: `VeracodeC#API.exe`

You can add `| more` or `| less` to the end of the command to enable scrolling through the output, depending on which commands are available in your environment.

The following images show the command output, which includes the supported calls, parameters, and the wrapper version.



If you only want to display the wrapper version, enter one of the following commands:

- For Java: `java -jar VeracodeJavaAPI.jar -wrapperversion`
- For C#: `VeracodeC#API.exe -wrapperversion`

API Wrapper Parameters


Most actions in the Veracode Java and C# API wrappers expose required or optional parameters. Some parameters apply to every action that the wrapper supports.



NOTE: Parameter names in the wrappers omit the underscores used in the APIs. For example, `app_id` in the API becomes `appid` in the wrapper.

The following table lists the parameters that apply to all actions:

Name	Type	Description
<code>action</code> Required	String	Name of the action.
<code>vid</code> Required	String	Veracode API ID.

Name	Type	Description
vkey	String	Veracode API key.
Required		
inputfilepath	String	Path to a CSV file that contains the parameters for the action. You can use <code>inputfilepath</code> to make multiple actions. In the CSV file, use one row to specify each action. For an example, see createuser.do .
logfilepath	String	Filepath of the log file where commands and messages are stored.
phost	String	Proxy host.  NOTE: Do not include the web protocol (http or https) with the proxy host. Including the protocol could cause the command to fail.
ppassword	String	Proxy password.
pport	String	Proxy port.
puser	String	Proxy user.

For example, run `getapplist` with these parameters to return a list of applications in the application portfolio of an account:

Using the Java wrapper:

```
java -jar VeracodeJavaAPI.jar -action getapplist -vid <Veracode API ID> -vkey <Veracode API key> -phost proxyhost.com -pport 8080 -puser <proxy username> -ppassword <proxy password>
```

Using the C# wrapper:

```
VeracodeC#API.exe -action getapplist -vid <Veracode API ID> -vkey <Veracode API key> -phost proxyhost.com -pport 8080 -puser <proxy username> -ppassword <proxy password>
```

To see the required and optional parameters for an action, you can run `-action` and review the returned text. The following images show the returned text when you run `-action createapp` in the Java or C# wrapper.



After reading the returned text, you know which parameters to include in your command. For example:

Using the Java wrapper:

```
java -jar VeracodeJavaAPI.jar -action createapp -appname <application-profile-name> -criticality <business-criticality> -vid <Veracode API ID> -vkey <Veracode API key>
```

Using the C# wrapper:

```
VeracodeC#API.exe -action createapp -appname <application-profile-name> -criticality <business-criticality> -vid <Veracode API ID> -vkey <Veracode API key>
```

You replace the parameter values with your Veracode API ID, Veracode API key, application name, and the level of business criticality of the application. See the [createapp.do](#) call for more information on the parameters.

Arguments

All parameters require an argument or value. For example, the `-criticality` parameter requires an argument of `VeryHigh`, `High`, `Medium`, `Low`, or `VeryLow`. When you run `-action`, to see the parameters for an action, you also see the list of arguments under Usage.

Simple Actions for the API Wrappers

A simple action is a single call to one of the Veracode XML APIs.

You can use the Veracode API wrappers to perform the following simple actions:

Upload Actions

- [createapp](#)
- [updateapp](#)
- [deleteapp](#)
- [getappinfo](#)
- [getapplist](#)
- [createbuild](#)
- [updatebuild](#)
- [deletebuild](#)
- [getbuildinfo](#)
- [getbuildlist](#)
- [uploadfile](#)
- [removefile](#)
- [getfilelist](#)
- [beginprescan](#)
- [getprescanresults](#)
- [beginscan](#)
- [getpolicylist](#)
- [getvendorlist](#)

Dynamic Scan Actions

- [*rescandynamicscan*](#)
- [*submitdynamicscan*](#)

Results Actions

- [*detailedreport*](#)
- [*summaryreport*](#)
- [*thirdpartyreport*](#)
- [*getappbuilds*](#)
- [*getcallstacks*](#)
- [*detailedreportpdf*](#)
- [*summaryreportpdf*](#)
- [*thirdpartyreportpdf*](#)



NOTE: To access the `detailedreportpdf`, `summaryreportpdf`, or `thirdpartyreportpdf` endpoints through the wrapper, you must call `detailedreport.do`, `summaryreport.do` or `thirdpartyreport.do`, using the `-format pdf` parameter.

Admin Actions

- [*createteam*](#)
- [*updateteam*](#)
- [*deleteteam*](#)
- [*getteamlist*](#)
- [*createuser*](#)
- [*updateuser*](#)
- [*deleteuser*](#)
- [*getuserinfo*](#)
- [*getuserlist*](#)
- [*getcurriculumlist*](#)
- [*gettracklist*](#)

Sandbox Actions

- [*createsandbox*](#)
- [*getsandboxlist*](#)
- [*updatesandbox*](#)
- [*promotesandbox*](#)
- [*deletesandbox*](#)

VAST Actions

- [*getsharedreportinfo*](#)
- [*getsharedreportlist*](#)
- [*sharedreport*](#)
- [*sharedreportpdf*](#)

Mitigation Actions

- [getmitigationinfo](#)
- [updatemitigationinfo](#)

Archer Actions

- [generatearcherreport](#)
- [downloadarcherreport](#)
- [archer](#)

Flaw Report Actions

- [generateflawreport](#)
- [downloadflawreport](#)

Composite Actions for the API Wrappers

A composite action corresponds to calls to multiple Veracode XML APIs to accomplish a specific workflow.

The Veracode API wrappers can perform the following composite actions:

- [uploadandscan](#)
- [createandsubmitdynamicrescan](#)
- [alldetailedreports](#)
- [passfail](#)
- [switchtosaml](#)

uploadandscan

The `uploadandscan` composite action enables you to upload files to the Veracode Platform for scanning.

The auto-scan option is always set to on with the `uploadandscan` call. See more information about the [auto-scan option](#).



NOTE: Parameter names in the wrappers omit the underscores used in the APIs. For example, `app_id` in the API becomes `appid` in the wrapper.

The `uploadandscan` call supports the following parameters:

Name	Type	Description
<code>appname</code> Required	String	Name of the application in the Veracode Platform.
<code>createprofile</code> Required	Boolean	Required if the specified application name is not in the Veracode Platform. Creates an application profile, if application name. Set to true to create a new profile.
<code>filepath</code> Required	String	Filepath or folderpath of the files you want to upload for scanning. By default, Veracode uploads files from the current directory.

Name	Type	Description
		this filepath. Ensure you have prepared the packaging guidance .
version Required	String	Name or version of the build that you want to scan.
createsandbox	Boolean	Creates a sandbox for the specified application.
criticality	String	Optional. Required if setting the createpolicy parameter. Criticality of the scan: <ul style="list-style-type: none"> • VeryHigh • High • Medium • Low • VeryLow
exclude	String	Case-sensitive, comma-separated list of module names to not scan as top-level modules. The ? wildcard matches 0 or more characters. The * wildcard matches 1 or more characters.
include	String	Case-sensitive, comma-separated list of module names to scan as top-level modules. The ? wildcard matches 0 or more characters. The * wildcard matches 1 or more characters.
pattern	String	Case-sensitive filename pattern that represents the file to save with a different name. The * wildcard matches exactly one character. The ? wildcard matches exactly one character. The numbered group that you can reference in the replacement pattern.
replacement	String	Replacement pattern that references groups in the filename pattern. For example, if the filename pattern is --\$1-\$2-\$3-\$4-\$5-\$6-\$7-\$8-\$9-\$10-\$11-\$12-\$13-\$14-\$15-\$16-\$17-\$18-\$19-\$20-\$21-\$22-\$23-\$24-\$25-\$26-\$27-\$28-\$29-\$30-\$31-\$32-\$33-\$34-\$35-\$36-\$37-\$38-\$39-\$40-\$41-\$42-\$43-\$44-\$45-\$46-\$47-\$48-\$49-\$50-\$51-\$52-\$53-\$54-\$55-\$56-\$57-\$58-\$59-\$60-\$61-\$62-\$63-\$64-\$65-\$66-\$67-\$68-\$69-\$70-\$71-\$72-\$73-\$74-\$75-\$76-\$77-\$78-\$79-\$80-\$81-\$82-\$83-\$84-\$85-\$86-\$87-\$88-\$89-\$90-\$91-\$92-\$93-\$94-\$95-\$96-\$97-\$98-\$99-\$100-\$101-\$102-\$103-\$104-\$105-\$106-\$107-\$108-\$109-\$110-\$111-\$112-\$113-\$114-\$115-\$116-\$117-\$118-\$119-\$120-\$121-\$122-\$123-\$124-\$125-\$126-\$127-\$128-\$129-\$130-\$131-\$132-\$133-\$134-\$135-\$136-\$137-\$138-\$139-\$140-\$141-\$142-\$143-\$144-\$145-\$146-\$147-\$148-\$149-\$150-\$151-\$152-\$153-\$154-\$155-\$156-\$157-\$158-\$159-\$160-\$161-\$162-\$163-\$164-\$165-\$166-\$167-\$168-\$169-\$170-\$171-\$172-\$173-\$174-\$175-\$176-\$177-\$178-\$179-\$180-\$181-\$182-\$183-\$184-\$185-\$186-\$187-\$188-\$189-\$190-\$191-\$192-\$193-\$194-\$195-\$196-\$197-\$198-\$199-\$200-\$201-\$202-\$203-\$204-\$205-\$206-\$207-\$208-\$209-\$210-\$211-\$212-\$213-\$214-\$215-\$216-\$217-\$218-\$219-\$220-\$221-\$222-\$223-\$224-\$225-\$226-\$227-\$228-\$229-\$230-\$231-\$232-\$233-\$234-\$235-\$236-\$237-\$238-\$239-\$240-\$241-\$242-\$243-\$244-\$245-\$246-\$247-\$248-\$249-\$250-\$251-\$252-\$253-\$254-\$255-\$256-\$257-\$258-\$259-\$260-\$261-\$262-\$263-\$264-\$265-\$266-\$267-\$268-\$269-\$270-\$271-\$272-\$273-\$274-\$275-\$276-\$277-\$278-\$279-\$280-\$281-\$282-\$283-\$284-\$285-\$286-\$287-\$288-\$289-\$290-\$291-\$292-\$293-\$294-\$295-\$296-\$297-\$298-\$299-\$300-\$301-\$302-\$303-\$304-\$305-\$306-\$307-\$308-\$309-\$310-\$311-\$312-\$313-\$314-\$315-\$316-\$317-\$318-\$319-\$320-\$321-\$322-\$323-\$324-\$325-\$326-\$327-\$328-\$329-\$330-\$331-\$332-\$333-\$334-\$335-\$336-\$337-\$338-\$339-\$340-\$341-\$342-\$343-\$344-\$345-\$346-\$347-\$348-\$349-\$350-\$351-\$352-\$353-\$354-\$355-\$356-\$357-\$358-\$359-\$360-\$361-\$362-\$363-\$364-\$365-\$366-\$367-\$368-\$369-\$370-\$371-\$372-\$373-\$374-\$375-\$376-\$377-\$378-\$379-\$380-\$381-\$382-\$383-\$384-\$385-\$386-\$387-\$388-\$389-\$390-\$391-\$392-\$393-\$394-\$395-\$396-\$397-\$398-\$399-\$400-\$401-\$402-\$403-\$404-\$405-\$406-\$407-\$408-\$409-\$410-\$411-\$412-\$413-\$414-\$415-\$416-\$417-\$418-\$419-\$420-\$421-\$422-\$423-\$424-\$425-\$426-\$427-\$428-\$429-\$430-\$431-\$432-\$433-\$434-\$435-\$436-\$437-\$438-\$439-\$440-\$441-\$442-\$443-\$444-\$445-\$446-\$447-\$448-\$449-\$450-\$451-\$452-\$453-\$454-\$455-\$456-\$457-\$458-\$459-\$460-\$461-\$462-\$463-\$464-\$465-\$466-\$467-\$468-\$469-\$470-\$471-\$472-\$473-\$474-\$475-\$476-\$477-\$478-\$479-\$480-\$481-\$482-\$483-\$484-\$485-\$486-\$487-\$488-\$489-\$490-\$491-\$492-\$493-\$494-\$495-\$496-\$497-\$498-\$499-\$500-\$501-\$502-\$503-\$504-\$505-\$506-\$507-\$508-\$509-\$510-\$511-\$512-\$513-\$514-\$515-\$516-\$517-\$518-\$519-\$520-\$521-\$522-\$523-\$524-\$525-\$526-\$527-\$528-\$529-\$530-\$531-\$532-\$533-\$534-\$535-\$536-\$537-\$538-\$539-\$540-\$541-\$542-\$543-\$544-\$545-\$546-\$547-\$548-\$549-\$550-\$551-\$552-\$553-\$554-\$555-\$556-\$557-\$558-\$559-\$560-\$561-\$562-\$563-\$564-\$565-\$566-\$567-\$568-\$569-\$570-\$571-\$572-\$573-\$574-\$575-\$576-\$577-\$578-\$579-\$580-\$581-\$582-\$583-\$584-\$585-\$586-\$587-\$588-\$589-\$590-\$591-\$592-\$593-\$594-\$595-\$596-\$597-\$598-\$599-\$600-\$601-\$602-\$603-\$604-\$605-\$606-\$607-\$608-\$609-\$610-\$611-\$612-\$613-\$614-\$615-\$616-\$617-\$618-\$619-\$620-\$621-\$622-\$623-\$624-\$625-\$626-\$627-\$628-\$629-\$630-\$631-\$632-\$633-\$634-\$635-\$636-\$637-\$638-\$639-\$640-\$641-\$642-\$643-\$644-\$645-\$646-\$647-\$648-\$649-\$650-\$651-\$652-\$653-\$654-\$655-\$656-\$657-\$658-\$659-\$660-\$661-\$662-\$663-\$664-\$665-\$666-\$667-\$668-\$669-\$670-\$671-\$672-\$673-\$674-\$675-\$676-\$677-\$678-\$679-\$680-\$681-\$682-\$683-\$684-\$685-\$686-\$687-\$688-\$689-\$690-\$691-\$692-\$693-\$694-\$695-\$696-\$697-\$698-\$699-\$700-\$701-\$702-\$703-\$704-\$705-\$706-\$707-\$708-\$709-\$710-\$711-\$712-\$713-\$714-\$715-\$716-\$717-\$718-\$719-\$720-\$721-\$722-\$723-\$724-\$725-\$726-\$727-\$728-\$729-\$730-\$731-\$732-\$733-\$734-\$735-\$736-\$737-\$738-\$739-\$740-\$741-\$742-\$743-\$744-\$745-\$746-\$747-\$748-\$749-\$750-\$751-\$752-\$753-\$754-\$755-\$756-\$757-\$758-\$759-\$760-\$761-\$762-\$763-\$764-\$765-\$766-\$767-\$768-\$769-\$770-\$771-\$772-\$773-\$774-\$775-\$776-\$777-\$778-\$779-\$780-\$781-\$782-\$783-\$784-\$785-\$786-\$787-\$788-\$789-\$790-\$791-\$792-\$793-\$794-\$795-\$796-\$797-\$798-\$799-\$800-\$801-\$802-\$803-\$804-\$805-\$806-\$807-\$808-\$809-\$810-\$811-\$812-\$813-\$814-\$815-\$816-\$817-\$818-\$819-\$820-\$821-\$822-\$823-\$824-\$825-\$826-\$827-\$828-\$829-\$830-\$831-\$832-\$833-\$834-\$835-\$836-\$837-\$838-\$839-\$840-\$841-\$842-\$843-\$844-\$845-\$846-\$847-\$848-\$849-\$850-\$851-\$852-\$853-\$854-\$855-\$856-\$857-\$858-\$859-\$860-\$861-\$862-\$863-\$864-\$865-\$866-\$867-\$868-\$869-\$870-\$871-\$872-\$873-\$874-\$875-\$876-\$877-\$878-\$879-\$880-\$881-\$882-\$883-\$884-\$885-\$886-\$887-\$888-\$889-\$890-\$891-\$892-\$893-\$894-\$895-\$896-\$897-\$898-\$899-\$900-\$901-\$902-\$903-\$904-\$905-\$906-\$907-\$908-\$909-\$910-\$911-\$912-\$913-\$914-\$915-\$916-\$917-\$918-\$919-\$920-\$921-\$922-\$923-\$924-\$925-\$926-\$927-\$928-\$929-\$930-\$931-\$932-\$933-\$934-\$935-\$936-\$937-\$938-\$939-\$940-\$941-\$942-\$943-\$944-\$945-\$946-\$947-\$948-\$949-\$950-\$951-\$952-\$953-\$954-\$955-\$956-\$957-\$958-\$959-\$960-\$961-\$962-\$963-\$964-\$965-\$966-\$967-\$968-\$969-\$970-\$971-\$972-\$973-\$974-\$975-\$976-\$977-\$978-\$979-\$980-\$981-\$982-\$983-\$984-\$985-\$986-\$987-\$988-\$989-\$990-\$991-\$992-\$993-\$994-\$995-\$996-\$997-\$998-\$999-\$1000-\$1001-\$1002-\$1003-\$1004-\$1005-\$1006-\$1007-\$1008-\$1009-\$1010-\$1011-\$1012-\$1013-\$1014-\$1015-\$1016-\$1017-\$1018-\$1019-\$1020-\$1021-\$1022-\$1023-\$1024-\$1025-\$1026-\$1027-\$1028-\$1029-\$1030-\$1031-\$1032-\$1033-\$1034-\$1035-\$1036-\$1037-\$1038-\$1039-\$1040-\$1041-\$1042-\$1043-\$1044-\$1045-\$1046-\$1047-\$1048-\$1049-\$1050-\$1051-\$1052-\$1053-\$1054-\$1055-\$1056-\$1057-\$1058-\$1059-\$1060-\$1061-\$1062-\$1063-\$1064-\$1065-\$1066-\$1067-\$1068-\$1069-\$1070-\$1071-\$1072-\$1073-\$1074-\$1075-\$1076-\$1077-\$1078-\$1079-\$1080-\$1081-\$1082-\$1083-\$1084-\$1085-\$1086-\$1087-\$1088-\$1089-\$1090-\$1091-\$1092-\$1093-\$1094-\$1095-\$1096-\$1097-\$1098-\$1099-\$1100-\$1101-\$1102-\$1103-\$1104-\$1105-\$1106-\$1107-\$1108-\$1109-\$1110-\$1111-\$1112-\$1113-\$1114-\$1115-\$1116-\$1117-\$1118-\$1119-\$1120-\$1121-\$1122-\$1123-\$1124-\$1125-\$1126-\$1127-\$1128-\$1129-\$1130-\$1131-\$1132-\$1133-\$1134-\$1135-\$1136-\$1137-\$1138-\$1139-\$1140-\$1141-\$1142-\$1143-\$1144-\$1145-\$1146-\$1147-\$1148-\$1149-\$1150-\$1151-\$1152-\$1153-\$1154-\$1155-\$1156-\$1157-\$1158-\$1159-\$1160-\$1161-\$1162-\$1163-\$1164-\$1165-\$1166-\$1167-\$1168-\$1169-\$1170-\$1171-\$1172-\$1173-\$1174-\$1175-\$1176-\$1177-\$1178-\$1179-\$1180-\$1181-\$1182-\$1183-\$1184-\$1185-\$1186-\$1187-\$1188-\$1189-\$1190-\$1191-\$1192-\$1193-\$1194-\$1195-\$1196-\$1197-\$1198-\$1199-\$1200-\$1201-\$1202-\$1203-\$1204-\$1205-\$1206-\$1207-\$1208-\$1209-\$1210-\$1211-\$1212-\$1213-\$1214-\$1215-\$1216-\$1217-\$1218-\$1219-\$1220-\$1221-\$1222-\$1223-\$1224-\$1225-\$1226-\$1227-\$1228-\$1229-\$1230-\$1231-\$1232-\$1233-\$1234-\$1235-\$1236-\$1237-\$1238-\$1239-\$1240-\$1241-\$1242-\$1243-\$1244-\$1245-\$1246-\$1247-\$1248-\$1249-\$1250-\$1251-\$1252-\$1253-\$1254-\$1255-\$1256-\$1257-\$1258-\$1259-\$1260-\$1261-\$1262-\$1263-\$1264-\$1265-\$1266-\$1267-\$1268-\$1269-\$1270-\$1271-\$1272-\$1273-\$1274-\$1275-\$1276-\$1277-\$1278-\$1279-\$1280-\$1281-\$1282-\$1283-\$1284-\$1285-\$1286-\$1287-\$1288-\$1289-\$1290-\$1291-\$1292-\$1293-\$1294-\$1295-\$1296-\$1297-\$1298-\$1299-\$1300-\$1301-\$1302-\$1303-\$1304-\$1305-\$1306-\$1307-\$1308-\$1309-\$1310-\$1311-\$1312-\$1313-\$1314-\$1315-\$1316-\$1317-\$1318-\$1319-\$1320-\$1321-\$1322-\$1323-\$1324-\$1325-\$1326-\$1327-\$1328-\$1329-\$1330-\$1331-\$1332-\$1333-\$1334-\$1335-\$1336-\$1337-\$1338-\$1339-\$1340-\$1341-\$1342-\$1343-\$1344-\$1345-\$1346-\$1347-\$1348-\$1349-\$1350-\$1351-\$1352-\$1353-\$1354-\$1355-\$1356-\$1357-\$1358-\$1359-\$1360-\$1361-\$1362-\$1363-\$1364-\$1365-\$1366-\$1367-\$1368-\$1369-\$1370-\$1371-\$1372-\$1373-\$1374-\$1375-\$1376-\$1377-\$1378-\$1379-\$1380-\$1381-\$1382-\$1383-\$1384-\$1385-\$1386-\$1387-\$1388-\$1389-\$1390-\$1391-\$1392-\$1393-\$1394-\$1395-\$1396-\$1397-\$1398-\$1399-\$1400-\$1401-\$1402-\$1403-\$1404-\$1405-\$1406-\$1407-\$1408-\$1409-\$1410-\$1411-\$1412-\$1413-\$1414-\$1415-\$1416-\$1417-\$1418-\$1419-\$1420-\$1421-\$1422-\$1423-\$1424-\$1425-\$1426-\$1427-\$1428-\$1429-\$1430-\$1431-\$1432-\$1433-\$1434-\$1435-\$1436-\$1437-\$1438-\$1439-\$1440-\$1441-\$1442-\$1443-\$1444-\$1445-\$1446-\$1447-\$1448-\$1449-\$1450-\$1451-\$1452-\$1453-\$1454-\$1455-\$1456-\$1457-\$1458-\$1459-\$1460-\$1461-\$1462-\$1463-\$1464-\$1465-\$1466-\$1467-\$1468-\$1469-\$1470-\$1471-\$1472-\$1473-\$1474-\$1475-\$1476-\$1477-\$1478-\$1479-\$1480-\$1481-\$1482-\$1483-\$1484-\$1485-\$1486-\$1487-\$1488-\$1489-\$1490-\$1491-\$1492-\$1493-\$1494-\$1495-\$1496-\$1497-\$1498-\$1499-\$1500-\$1501-\$1502-\$1503-\$1504-\$1505-\$1506-\$1507-\$1508-\$1509-\$1510-\$1511-\$1512-\$1513-\$1514-\$1515-\$1516-\$1517-\$1518-\$1519-\$1520-\$1521-\$1522-\$1523-\$1524-\$1525-\$1526-\$1527-\$1528-\$1529-\$1530-\$1531-\$1532-\$1533-\$1534-\$1535-\$1536-\$1537-\$1538-\$1539-\$1540-\$1541-\$1542-\$1543-\$1544-\$1545-\$1546-\$1547-\$1548-\$1549-\$1550-\$1551-\$1552-\$1553-\$1554-\$1555-\$1556-\$1557-\$1558-\$1559-\$1560-\$1561-\$1562-\$1563-\$1564-\$1565-\$1566-\$1567-\$1568-\$1569-\$1570-\$1571-\$1572-\$1573-\$1574-\$1575-\$1576-\$1577-\$1578-\$1579-\$1580-\$1581-\$1582-\$1583-\$1584-\$1585-\$1586-\$1587-\$1588-\$1589-\$1590-\$1591-\$1592-\$1593-\$1594-\$1595-\$1596-\$1597-\$1598-\$1599-\$1600-\$1601-\$1602-\$1603-\$1604-\$1605-\$1606-\$1607-\$1608-\$1609-\$1610-\$1611-\$1612-\$1613-\$1614-\$1615-\$1616-\$1617-\$1618-\$1619-\$1620-\$1621-\$1622-\$1623-\$1624-\$1625-\$1626-\$1627-\$1628-\$1629-\$1630-\$1631-\$1632-\$1633-\$1634-\$1635-\$1636-\$1637-\$1638-\$1639-\$1640-\$1641-\$1642-\$1643-\$1644-\$1645-\$1646-\$1647-\$1648-\$1649-\$1650-\$1651-\$1652-\$1653-\$1654-\$1655-\$1656-\$1657-\$1658-\$1659-\$1660-\$1661-\$1662-\$1663-\$1664-\$1665-\$1666-\$1667-\$1668-\$1669-\$1670-\$1671-\$1672-\$1673-\$1674-\$1675-\$1676-\$1677-\$1678-\$1679-\$1680-\$1681-\$1682-\$1683-\$1684-\$1685-\$1686-\$1687-\$1688-\$1689-\$1690-\$1691-\$1692-\$1693-\$1694-\$1695-\$1696-\$1697-\$1698-\$1699-\$1700-\$1701-\$1702-\$1703-\$1704-\$1705-\$1706-\$1707-\$1708-\$1709-\$1710-\$1711-\$1712-\$1713-\$1714-\$1715-\$1716-\$1717-\$1718-\$1719-\$1720-\$1721-\$1722-\$1723-\$1724-\$1725-\$1726-\$1727-\$1728-\$1729-\$1730-\$1731-\$1732-\$1733-\$1734-\$1735-\$1736-\$1737-\$1738-\$1739-\$1740-\$1741-\$1742-\$1743-\$1744-\$1745-\$1746-\$1747-\$1748-\$1749-\$1750-\$1751-\$1752-\$1753-\$1754-\$1755-\$1756-\$1757-\$1758-\$1759-\$1760-\$1761-\$1762-\$1763-\$1764-\$1765-\$1766-\$1767-\$1768-\$1769-\$1770-\$1771-\$1772-\$1773-\$1774-\$1775-\$1776-\$1777-\$1778-\$1779-\$1780-\$1781-\$1782-\$1783-\$1784-\$1785-\$1786-\$1787-\$1788-\$1789-\$1790-\$1791-\$1792-\$1793-\$1794-\$1795-\$1796-\$1797-\$1798-\$1799-\$1800-\$1801-\$1802-\$1803-\$1804-\$1805-\$1806-\$1807-\$1808-\$1809-\$1810-\$1811-\$1812-\$1813-\$1814-\$1815-\$1816-\$1817-\$1818-\$1819-\$1820-\$1821-\$1822-\$1823-\$1824-\$1825-\$1826-\$1827-\$1828-\$1829-\$1830-\$1831-\$1832-\$1833-\$1834-\$1835-\$1836-\$1837-\$1838-\$1839-\$1840-\$1841-\$1842-\$1843-\$1844-\$1845-\$1846-\$1847-\$1848-\$1849-\$1850-\$1851-\$1852-\$1853-\$1854-\$1855-\$1856-\$1857-\$1858-\$1859-\$1860-\$1861-\$1862-\$1863-\$1864-\$1865-\$1866-\$1867-\$1868-\$1869-\$1870-\$1871-\$1872-\$1873-\$1874-\$1875-\$1876-\$1877-\$1878-\$1879-\$1880-\$1881-\$1882-\$1883-\$1884-\$1885-\$1886-\$1887-\$1888-\$1889-\$1890-\$1891-\$1892-\$1893-\$1894-\$1895-\$1896-\$1897-\$1898-\$1899-\$1900-\$1901-\$1902-\$1903-\$1904-\$1905-\$1906-\$1907-\$1908-\$1909-\$1910-\$1911-\$1912-\$1913-\$1914-\$1915-\$1916-\$1917-\$1918-\$1919-\$1920-\$1921-\$1922-\$1923-\$1924-\$1925-\$1926-\$1927-\$1928-\$1929-\$1930-\$1931-\$1932-\$1933-\$1934-\$1935-\$1936-\$1937-\$1938-\$1939-\$1940-\$1941-\$1942-\$1943-\$1944-\$1945-\$1946-\$1947-\$1948-\$1949-\$1950-\$1951-\$1952-\$1953-\$1954-\$1955-\$1956-\$1957-\$1958-\$1959-\$1960-\$1961-\$1962-\$1963-\$1964-\$1965-\$1966-\$1967-\$1968-\$1969-\$1970-\$1971-\$1972-\$1973-\$1974-\$1975-\$1976-\$1977-\$1978-\$1979-\$1980-\$1981-\$1982-\$1983-\$1984-\$1985-\$1986-\$1987-\$1988-\$1989-\$1990-\$1991-\$1992-\$1993-\$1994-\$1995-\$1996-\$1997-\$1998-\$1999-\$2000-\$2001-\$2002-\$2003-\$2004-\$2005-\$2006-\$2007-\$2008-\$2009-\$2010-\$2011-\$2012-\$2013-\$2014-\$2015-\$2016-\$2017-\$2018-\$2019-\$2020-\$2021-\$2022-\$2023-\$2024-\$2025-\$2026-\$2027-\$2028-\$2029-\$2030-\$2031-\$2032-\$2033-\$2034-\$2035-\$2036-\$2037-\$2038-\$2039-\$2040-\$2041-\$2042-\$2043-\$2044-\$2045-\$2046-\$2047-\$2048-\$2049-\$2050-\$2051-\$2052-\$2053-\$2054-\$2055-\$2056-\$2057-\$2058-\$2059-\$2060-\$2061-\$2062-\$2063-\$2064-\$2065-\$2066-\$2067-\$2068-\$2069-\$2070-\$2071-\$2072-\$2073-\$2074-\$2075-\$2076-\$2077-\$2078-\$2079-\$2080-\$2081-\$2082-\$2083-\$2084-\$2085-\$2086-\$2087-\$2088-\$2089-\$2090-\$2091-\$2092-\$2093-\$2094-\$2095-\$2096-\$2097-\$2098-\$2099-\$2100-\$2101-\$2102-\$2103-\$2104-\$2105-\$2106-\$2107-\$2108-\$2109-\$2110-\$2111-\$2112-\$2113-\$2114-\$2115-\$2116-\$2117-\$2118-\$2119-\$2120-\$2121-\$2122-\$2123-\$2124-\$2125-\$2126-\$2127-\$2128-\$2129-\$2130-\$2131-\$2132-\$2133-\$2134-\$2135-\$2136-\$2137-\$2138-\$2139-\$2140-\$2141-\$2142-\$2143-\$2144-\$2145-\$2146-\$2147-\$2148-\$2149-\$2150-\$2151-\$2152-\$2153-\$2154-\$2155-\$2156-\$2157-\$2158-\$2159-\$2160-\$2161-\$2162-\$2163-\$2164-\$2165-\$2166-\$2167-\$2168-\$2169-\$2170-\$2171-\$2172-\$2173-\$2174-\$2175-\$2176-\$2177-\$2178-\$2179-\$2180-\$2181-\$2182-\$2183-\$2184-\$2185-\$2186-\$2187-\$2188-\$2189-\$2190-\$2191-\$2192-\$2193-\$2194-\$2195-\$2196-\$2197-\$2198-\$2199-\$2200-\$2201-\$2202-\$2203-\$2204-\$2205-\$2206-\$2207-\$2208-\$2209-\$2210-\$2211-\$2212-\$2213-\$2214-\$2215-\$2216-\$2217-\$2218-\$2219-\$2220-\$2221-\$2222-\$2223-\$2224-\$2225-\$2226-\$2227-\$2228-\$2229-\$2230-\$2231-\$2232-\$2233-\$2234-\$2235-\$2236-\$2237-\$2238-\$2239-\$2240-\$2241-\$2242-\$2243-\$2244-\$2245-\$2246-\$2247-\$2248-\$2249-\$2250-\$2251-\$2252-\$2253-\$2254-\$2255-\$2256-\$2257-\$2258-\$2259-\$2260-\$2261-\$2262-\$2263-\$2264-\$2265-\$2266-\$2267-\$2268-\$2269-\$2270-\$2271-\$2272-\$2273-\$2274-\$2275-\$2276-\$2277-\$2278-\$2279-\$2280-\$2281-\$2282-\$2283-\$2284-\$2285-\$2286-\$2287-\$2288-\$2289-\$2290-\$2291-\$2292-\$2293-\$2294-\$2295-\$2296-\$2297-\$2298-\$2299-\$2300-\$2301-\$2302-\$2303-\$2304-\$2305-\$2306-\$2307-\$2308-\$2309-\$2310-\$2311-\$2312-\$2313-\$2314-\$2315-\$2316-\$2317-\$2318-\$2319-\$2320-\$2321-\$2322-\$2323-\$2324-\$2325-\$2326-\$2327-\$2328-\$2329-\$2330-\$2331-\$2332-\$2333-\$2334-\$2335-\$2336-\$2337-\$2338-\$2339-\$2340-\$2341-\$2342-\$2343-\$2344-\$2345-\$2346-\$2347-\$2348-\$2349-\$2350-\$2351-\$2352-\$2353-\$2354-\$2355-\$2356-\$2357-\$2358-\$2359-\$2360-\$2361-\$2362-\$2363-\$2364-\$2365-\$2366-\$2367-\$2368-\$2369-\$2370-\$2371-\$2372-\$2373-\$2374-\$2375-\$2376-\$2377-\$2378-\$2379-\$2380-\$2381-\$2382-\$2383-\$2384-\$2385-\$2386-\$2387-\$2388-\$2389-\$2390-\$2391-\$2392-\$2393-\$2394-\$2395-\$2396-\$2397-\$2398-\$2399-\$2400-\$2401-\$2402-\$2403-\$2404-\$2405-\$2406-\$2407-\$2408-\$2409-\$2410-\$2411-\$2412-\$2413-\$2414-\$2415-\$2416-\$2417-\$2418-\$2419-\$2420-\$2421-\$2422-\$2423-\$2424-\$2425-\$2426-\$2427-\$2428-\$2429-\$2430-\$2431-\$2432-\$2433-\$2434-\$2435-\$2436-\$2437-\$2438-\$2439-\$2440-\$2441-\$2442-\$2443-\$2444-\$2445-\$2446-\$2447-\$2448-\$2449-\$2450-\$2451-\$2452-\$2453-\$2454-\$2455-\$2456-\$2457-\$2458-\$2459-\$2460-\$2461-\$2462-\$2463-\$2464-\$2465-\$2466-\$2467-\$2468-\$2469-\$2470-\$2471-\$2472-\$2473-\$2474-\$2475-\$2476-\$2477-\$2478-\$2479-\$2480-\$2481-\$2482-\$2483-\$2484-\$2485-\$2486-\$2487-\$2488-\$2489-\$2490-\$2491-\$2492-\$2493-\$2494-\$2495-\$2496-\$2497-\$2498-\$2499-\$2500-\$2501-\$2502-\$2503-\$2504-\$2505-\$2506-\$2507-\$2508-\$2509-\$2510-\$2511-\$2512-\$2513-\$2514-\$2515-\$2516-\$2517-\$2518-\$2519-\$2520-\$2521-\$2522-\$2523-\$2524-\$2525-\$2526-\$2527-\$2528-\$2529-\$2530-\$2531-\$2532-\$2533-\$2534-\$2535-\$2536-\$2537-\$2538-\$2539-\$2540-\$2541-\$2542-\$2543-\$2544-\$2545-\$2546-\$2547-\$2548-\$2549-\$2550-\$2551-\$2552-\$2553-\$2554-\$2555-\$2556-\$2557-\$2558-\$2559-\$2560-\$2561-\$2562-\$2563-\$2564-\$2565-\$2566-\$2567-\$2568-\$2569-\$2570-\$2571-\$2572-\$2573-\$2574-\$2575-\$2576-\$2577-\$2578-\$2579-\$2580-\$2581-\$2582-\$2583-\$2584-\$2585-\$2586-\$2587-\$2588-\$2589-\$2590-\$2591-\$2592-\$2593-\$2594-\$2595-\$2596-\$2597-\$2598-\$25

Name	Type	Description
		Veracode recommends that you use the t ensure the scan completes even though th unsupported frameworks.

Examples

The following example commands use the `uploadandscan` action in specific scenarios:

- Create an application and start a sandbox scan.

```
java -jar VeracodeJavaAPI.jar -action uploadandscan -vid
<VeracodeApiId> -vkey <VeracodeApiKey> -appname myapp -
createprofile true -teams myteam -criticality VeryHigh -
sandboxname mysandbox -createsandbox true -version <unique
version> -filepath /workspace/myapp.jar
```

```
VeracodeC#API -action uploadandscan -vid <VeracodeApiId> -vkey
<VeracodeApiKey> -appname myapp -createprofile true -teams myteam -
criticality VeryHigh -sandboxname mysandbox -createsandbox true -
version <unique version> -filepath /workspace/myapp.jar
```

- Create an application, start a sandbox scan that only includes modules selected in the previous scan, and wait 30 minutes for the scan to complete.

```
java -jar VeracodeJavaAPI.jar -action uploadandscan -vid
<VeracodeApiId> -vkey <VeracodeApiKey> -appname myapp -
createprofile true -teams myteam -criticality VeryHigh -
sandboxname mysandbox -createsandbox true -version <unique
version> -scantimeout 30 -selectedpreviously true -filepath /
workspace/myapp.jar
```

```
VeracodeC#API -action uploadandscan -vid <VeracodeApiId> -vkey
<VeracodeApiKey> -appname myapp -createprofile true -teams myteam -
criticality VeryHigh -sandboxname mysandbox -createsandbox true -
version <unique version> -scantimeout 30 -selectedpreviously true -
filepath /workspace/myapp.jar
```

createandsubmitdynamicrescan

The `createandsubmitdynamicrescan` composite action enables you to create and submit a DynamicDS rescan on the Veracode Platform.



NOTE: Parameter names in the wrappers omit the underscores used in the APIs. For example, `app_id` in the API becomes `appid` in the wrapper.

This action supports the following parameters:

Name	Type	Description
appname	String	Application name.
Required		

Name	Type	Description
<code>flawonly</code>	Boolean	Default is false. Set this parameter to true if you want to skip the previous scan.
<code>starttime</code>	String	Default is immediately. Enter a string for the start date and time.
<code>endtime</code>	String	Default is one day after the start time. Enter a string for the end date and time.
<code>version</code>	String	Name or version of the build that you want to scan.

alldetailedreports

The `alldetailedreports` composite action downloads the detailed report for the latest build of each application in the user account.



NOTE: Parameter names in the wrappers omit the underscores used in the APIs. For example, `app_id` in the API becomes `appid` in the wrapper.

This action supports the following parameters:

Name	Type	Description
<code>outputfolderpath</code>	String	Directs the report to the specified folder.
Required		
<code>includeinprogress</code>	Boolean	When true, includes build data for builds with a status of in progress.
<code>onlylatest</code>	Boolean	When false, includes build data for previous builds.
<code>reportchangedsince</code>	Boolean	Includes build data only for builds with scan reports that have changed since the specified date.

passfail

The `passfail` composite action retrieves the latest detailed report for the application with the specified `appid` parameter. It prints to the console the name of the submitter, the latest build, and the policy compliance status of the scan.

switchtosaml

The `switchtosaml` composite action calls `getuserlist.do`. It searches users on the list whose `is_saml_user` parameter is not set to true and whose `login_account_type` parameter is not `api`. The action then changes the `is_saml_user` parameter to true for those users. The parameter `customid` or `username` is required.

Using the API Wrappers as a Library or a Command-Line Application

When used as a library, the Veracode API wrapper enables programmers to no longer have to programmatically set up web requests and look up the URL and query string parameters of the Veracode API functions they want to call. When used as a command-line application, the Veracode API wrapper allows users to perform single and composite actions that require no programming knowledge. Read more about:

- [Using the API wrappers from the Command Line](#)
- [Referencing the API wrappers from Visual Studio](#)
- [Referencing the API wrappers from Eclipse](#)

Referencing the Veracode C# API Wrapper from Visual Studio

To reference the Veracode C# API wrapper from your Visual Studio project:

- 1 In Visual Studio, select **View > Solution Explorer**.
- 2 Right-click the **References** folder and select **Add Reference**.



- 3 In the Add Reference dialog, select the **Browse** tab.
- 4 Locate and select the `VeracodeC#API.exe` file.



- 5 Import the `com.veracode.apiwrapper` namespace.



- 6 Optionally, complete the following steps to display the API wrapper documentation in the Visual Studio interface. The documentation XML file must be in the same directory as the `VeracodeC#API.exe` file.
 - a In Solution Explorer, expand the **References** folder.
 - b Double-click **VeracodeC#API.exe**.



Referencing the Veracode Java API Wrapper from Eclipse

To reference the Veracode Java API wrapper from your Eclipse project:

- 1 In Eclipse, select **Window > Show View > Package Explorer**.
- 2 Right-click your project folder and select **Build Path > Configure Build Path**.



- 3 Select the **Libraries** tab.
- 4 Select **Add External JARs...**
- 5 Locate and select the `VeracodeJavaAPI.jar` file.



- 6 Click **Open**.
- 7 Import the `com.veracode.apiwrapper.wrappers` package.



8 Optionally, complete the following steps to display the API wrapper documentation in the Eclipse interface.

- a In the Configure Build Path window, select the **Libraries** tab.
- b Expand the **VeracodeJavaAPI.jar** node and select **Javadoc Location: (None)**.



- c Click **Edit...**
- d Browse to the folder containing the documentation and, then, click **OK**.

Have a Question??

Engage with the Veracode Community

You can connect with peers and Veracode subject matter experts in the Veracode Community for support, to learn best practices, and to collaborate on how Veracode products and services can work better for you.

To access the Veracode Community, navigate to community.veracode.com.



Accessing the Community

The Veracode Community is a public site. To engage with community members, you are asked to log in. If you have not registered for the Veracode Community, you can create a profile from **Login > Not a member ?** where you are prompted to enter basic user information. After you register, you are sent an email providing account instructions and how to set up your password. After your account is registered, you can log in from the button on the top-right of the home page.

Community Features

From the home page, you can navigate to the following different pages:

- Getting Started** Read about the features available on the Veracode Community, and how to get started with your Veracode program.
- Groups** Join a group for discussions on Veracode products and application security topics on the [Groups](#) page. You can discuss trends and gain insights, grouped by topic area.
- Education & Training** Watch Veracode AppSec Tutorials to learn more about application security on the [Education and Training](#) page. You can learn about topics such as flaw remediation, scanning with Veracode, and how to integrate with Veracode in your software development lifecycle.
- Integrations** Download Veracode integrations for APIs and plugins, and read more about how to integrate with Veracode throughout your entire software development lifecycle from the [Integrations](#) page.

Ideas Submit your ideas for how Veracode products and services can work better in your environment from the [Ideas](#) page. You can also vote and comment on existing ideas.



NOTE: This feature is available to Veracode customers only.

Search Use the search bar at the top of each page to search across product documentation, knowledge articles, blog posts, as well as questions and answers posted to the Veracode Community. The search bar models standard search behavior.

My Activity You can view a summary of your activity on the Veracode Community from the **My Activity** page. If you open a support case, you can also read the status of your case.

Demo Request button Click the **Demo Request** button to schedule a demonstration or on-boarding call from Veracode. Your program manager contacts you directly after you provide your availability.

Ask Community button If you cannot find the information you are looking for from the content available in the Community, click the **Ask Community** button to pose a question. You can receive input from fellow Veracode Community members.

Contact Support You can contact Veracode Technical Support if you have an urgent request by clicking **Contact Support** from the login menu.

Veracode Blog

Read the [Veracode Blog](#) for articles on the latest application security research, news, and education.

Copyright 2020 VERACODE, All Rights Reserved

[Privacy Policy](#)

[Terms of Use](#)