

Cylance Engine

Integration Guide

1.2

Contents

- What is the Cylance Engine?..... 5**
 - Data Flow: Analyzing a file with the Cylance Engine..... 5

- How the Cylance Engine analyzes a file..... 6**
 - Scoring files with Cylance AI..... 6
 - Scoring and threat indicators..... 6
 - Score generated by the Cylance Engine..... 6
 - Role of threat indicators in scoring..... 7
 - Use of centroids in the Cylance Engine..... 8
 - Restricted and allowed list of file hashes..... 8

- System requirements for the Cylance Engine..... 9**
 - Hardware requirements..... 9
 - Supported operating systems..... 9
 - Requirements: Microsoft .NET..... 10
 - Requirements: Mono..... 10
 - Requirements: Python..... 10
 - Requirements: Multiple instances of the Cylance Engine on one machine..... 10

- Installing and updating the Cylance Engine..... 12**
 - Install the Cylance Engine on a Linux distribution..... 12
 - Query the version of your Cylance Engine on a Linux distribution..... 12
 - Update the version of your Cylance Engine on a Linux distribution..... 13
 - Remove the Cylance Engine from a Linux distribution..... 13
 - Install the Cylance Engine on a Windows distribution..... 13
 - Query the version of your Cylance Engine on a Windows distribution..... 14
 - Update the version of your Cylance Engine on a Windows distribution..... 14
 - Remove the Cylance Engine from a Windows distribution..... 14

- File-scoring service..... 15**
 - Sample scored service script..... 15
 - Sentinel file..... 15
 - Environment variables..... 16
 - Command-line options for the Cylance Engine..... 16
 - Configuration file for the Cylance Engine..... 18

- File-scoring service protocols..... 27**
 - Cylance RESTful API..... 27
 - Getting model details..... 28
 - Scoring a file..... 30
 - Explaining the score for a file..... 36

Shutting down the service.....	37
Password-protected archives.....	38

Appendix: Cylance Infinity Data Service..... 39

Authentication of requests.....	39
Response status codes.....	39
Service endpoints.....	40
Centroids endpoint.....	40
Wblist endpoint.....	43

Appendix: Threat indicators..... 44

Anomalies.....	44
Collection.....	48
Data loss.....	50
Deception.....	51
Destruction.....	55
Shellcodes.....	58
Miscellaneous indicators.....	59

Appendix: Prometheus monitoring support..... 61

Appendix: CylanceTcpService Protocol..... 63

Process command.....	63
Shutdown command.....	65
Multiple scores for a file.....	65
Classless-based and activity-class-based scoring.....	65
Passwords specified for archives.....	66
File-scoring applications.....	66
Samplescore client.....	66
Ttmstatic script.....	67
InfinityDaemonClient utility.....	68

Legal notice..... 73

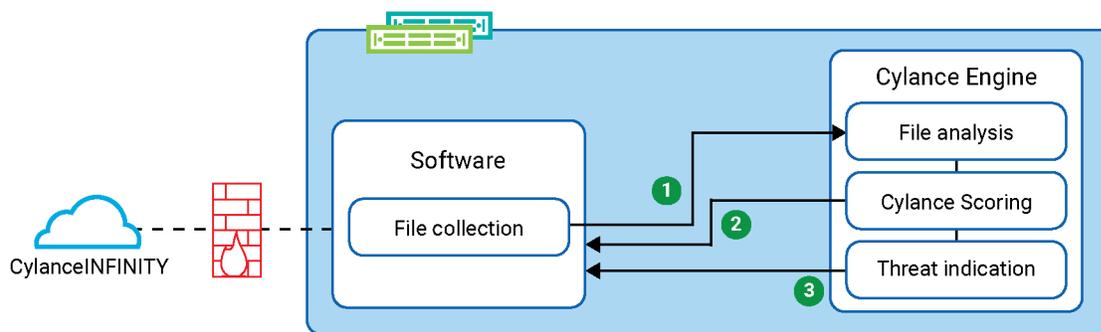
What is the Cylance Engine?

The Cylance Engine is used to discover and classify malware using Cylance AI, the industry's longest running, continuously improving, predictive AI in market.

The Cylance Engine can provide two types of information about a file:

- A file score in a range from -1.0 (malicious) to +1.0 (benign)
- A set of threat indicators that helps to identify the characteristics of the file that contribute to its score

Data Flow: Analyzing a file with the Cylance Engine



When the Cylance Engine analyzes a file, the following actions occur:

1. A file is sent to the Cylance Engine for analysis.
2. The Cylance Engine analyzes the file and returns a score.
3. Optionally, the Cylance Engine can also return a set of threat indicators.

How the Cylance Engine analyzes a file

To analyze a file, the Cylance Engine uses Cylance AI, a set of threat indicators, centroids, and a restricted and allowed list of file hashes.

Scoring files with Cylance AI

The Cylance Engine uses Cylance AI to classify files as bad or good with a certain level of confidence. The classification is made based on machine learning models. The models are being constantly improved using machine learning to refine the accuracy of the results.

The models contained in this version of the Cylance Engine are listed in the table below. Each of these models requires a corresponding dynamic library (a shared object in Linux terms).

File type	Model name	Extensions
Windows executables	Ensemble-20230818-S3V3-PE7E.cym	.acm, .ax, .cpl, .drv, .efi, .mui, .ocx, .src, .sys, .tsp, .exe, .dll
macOS executables	Ensemble-20210721-S3V4-MO3.cym Ensemble-20210409-S0V2-MOFAT.cym	(none), .o, .dylib, .bundle
Linux executables	Ensemble-20180730-S2V7-ELF2.cym	(none), .o, .ko, .mod, .so
OLE files	Ensemble-20180718-S3V3-OLE3.cym	.doc, .xls, .ppt
OOXML files	Ensemble-20180718-S3V3-OOXML3.cym	.docx, .xlsx, .pptx
PDF files	Ensemble-20230607-S3-PDF4.cym	.pdf
Archive files	Ensemble-20190319-S0V5-ARC.cym	.zip, .7z, .rar, .tar, .gz, .bz2, .xz Note: Files with the .zip extension are supported whether they are password-protected or not. Files with the .rar extension are supported only if they are not password-protected.

Scoring and threat indicators

A client can access two types of information from the Cylance Engine:

- A file score in a range from -1.0 (malicious) to +1.0 (benign).
- A set of threat indicators that helps to identify the characteristics of the file that contribute to its score.

Score generated by the Cylance Engine

The score represents the confidence the Cylance Engine has that a file is good or bad. Scores returned by the Cylance Engine are decimal numbers ranging from -1.0 to +1.0. A negative value indicates that the file has been

classified as a potentially bad file and a positive value indicates that the file has been classified as a potentially good file. The numerical value indicates the confidence in this file being identified as a risk or not.

For example:

Score	Meaning
-0.6	There is a 60% confidence level that the file is bad.
+0.6	There is a 60% confidence level that the file is good.

Note: The numerical value represents the confidence in a file being identified as a risk or not; it is not an indication of the harmfulness of a file.

If an error occurs during the processing of the file, the score may be NaN (not a number) to indicate that a score was not generated. In this case, the result should contain more information on the error.

Suggested implementation

As a guideline, the following conventions serve to classify the files.

Score	Meaning
Less than -0.6	Malware
-0.6 to 0	Suspicious
Greater than 0	Safe

Scoring considers not only the threat indicators, but also the relationships between these and many other characteristics of the file. Because the scoring uses complex mathematical models that operate directly on the features of the file, it is not possible to determine the score only from the threat indicators.

Role of threat indicators in scoring

Threat indicators are observations about a file or archive that the Cylance Engine has analyzed. These indicators help analysts better understand why the Cylance Engine has identified a file as a risk. They provide insight into potential abuse in a quick and easy-to-use format.

It is important to note that there are legitimate uses for each of the identified indicators. The existence of an indicator is not proof positive that an object is acting in a malicious manner. For example, if the file is a process debugger, it may have legitimate use of SEDebugPriv or process injection. Software installers frequently bundle an executable inside.

It is also important to note that these are specific indicators that have a high prevalence in malware, but they do not represent the machine-learning models that we use for classifying a file as good or bad. These models measure millions of data points, and while some of these data points correspond to these specific indicators, the final score for a file is determined by a complex synthesis of all data points. This limited set of threat indicators exists specifically because machine-learning models are difficult for humans to reason about.

Each indicator defines an area that has been frequently seen in malicious software. Many indicators represent capabilities of the included binary; other indicators represent attempts at deception. Each indicator has been identified as a frequent and strongly indicative feature based on a deep analysis of over 100 million binaries.

Threat indicators are grouped into categories to aid in context. Categories help to identify certain potentially undesirable or malicious capabilities.

Note: Threat indicators are available for all supported file types except for the ELF file type.

For a complete list of the threat indicators exposed and a brief description of each indicator, see [Appendix: Threat indicators](#).

Use of centroids in the Cylance Engine

Centroids are used to adjust the classification of a group of files by Cylance AI between updates to the Cylance Engine. Centroids are produced whenever an adjustment is deemed necessary.

The Cylance Engine can read updated centroids in one of two ways. The first method is to read a file stored in a local file location; the Cylance Engine periodically checks this location to see if the file has been updated. These centroids must be downloaded manually as described in the [Appendix: Cylance Infinity Data Service](#). Centroids downloaded manually include all released centroids for that model.

The second method is to retrieve automatically only the centroids that the Cylance Engine does not already have, whether they shipped with the Cylance Engine or were downloaded later, based on a manifest that contains a list of centroids for each of the models. Retrieving the manifest is a lightweight operation. The Cylance Engine then uses the manifest to download any centroids that it does not already have. The set of centroids downloaded in this manner may not match those retrieved via the [Infinity Public Data API](#) for distribution-efficiency reasons. For more information, see the ManifestCentroidUpdate section in [Configuration file for the Cylance Engine](#).

Note: On Mono, the Amazon Root CA 1 certificate might not be installed automatically. This certificate is required to retrieve the manifest-based centroids. You can download the certificate from <https://www.amazontrust.com/repository/AmazonRootCA1.pem> and import it into the Mono certificate store using the cert-sync utility that ships with Mono.

Restricted and allowed list of file hashes

The Cylance Engine can maintain a list of SHA256 file hashes that bypasses the machine-learning scoring algorithm. If a file hash is present in the allowed list, the corresponding file is assigned a good score (1.0); if the file is present in the restricted list, the file is assigned a bad score (-1.0).

BlackBerry also maintains a global restricted and allowed list of file hashes that have been determined to be bad or good by additional qualification means outside of the machine-learning algorithm. You can download this list using the [Infinity Public Data API](#).

System requirements for the Cylance Engine

To get started setting up the Cylance Engine, review this section and verify that your organization's environment meets the requirements.

Hardware requirements

The Cylance Engine requires Intel or AMD 64-bit processors. For other architectures, contact BlackBerry customer support. The minimum system requirements for the Cylance Engine are the same as the minimum system requirements for the operating system.

Supported operating systems

Supported OS: Linux distributions

The Cylance Engine supports several Linux x64 distributions. This includes service scripts, manual pages, and a sample client, all packaged into a format that the Linux distribution package manager supports. A .zip package is also available that can be used without a distribution package manager. The following table lists the current set of supported Linux distributions.

Operating system	Minimum version	Tested version
Debian	9	9, 10, 11, 12
Ubuntu	16.04	16.04, 18.04, 20.04, 22.04
CentOS	8	8
Red Hat	8	8, 9
Fedora	28	28, 33, 39

Supported OS: Windows Server versions

The Cylance Engine supports the following Windows Server x64 versions.

- Windows Server 2022 (Standard, Datacenter, and Server Core)
- Windows Server 2019 (Standard, Datacenter, and Server Core)
- Windows Server 2016 (Standard, Datacenter, Essentials, and Server Core)
- Windows Server 2012 R2 (Standard, Datacenter, Essentials, Server Core, Embedded, and Foundation)
- Windows Server 2012 (Standard, Datacenter, Essentials, Server Core, Embedded, and Foundation)

Windows support is also provided as a .zip package only with no installer. For more information, see [Install the Cylance Engine on a Windows distribution](#).

Requirements: Microsoft .NET

Starting with the Cylance Engine version 1.2, the Cylance Engine has a package that uses the Microsoft .NET 8.0 runtime for cross-platform compatibility. The package includes the required runtime files. For more information about Microsoft ASP.NET Core, visit the [Microsoft .NET website](#).

The Microsoft .NET Framework runtime packages depend on `libcicu` and `libssl` but are not listed as package dependencies because their names differ on different Linux distributions. For the exact package names, see <https://github.com/dotnet/core/blob/master/Documentation/linux-prereqs.md>. In the case that any required libraries are missing, `CylanceTcpService` reports the missing library and exits.

Requirements: Mono

Note: The Mono versions of the Cylance Engine installation packages are no longer actively maintained.

The Cylance Engine is natively built in C# and can also use Mono for cross-platform support, although it is highly recommended that Microsoft .NET be used instead.

The Cylance Engine installation package comes with Mono, so there is no need to install Mono before you install the Cylance Engine.

The Mono implementation included with the Cylance Engine installation package has been tailored to the needs of the Cylance Engine and is based on version 6.4.0. To reduce size and improve portability, any libraries or assemblies not required have been removed. This modified version is self-contained, does not modify any system configurations, and causes no issues with existing Mono installations. For more information about Mono, visit the [Mono Project website](#).

Requirements: Python

The Cylance Engine requires the installation of Python 2.7 or greater to use the included commands `ttmstatic` and `samplescore`. Python 3.7 or greater is recommended.

Requirements: Multiple instances of the Cylance Engine on one machine

You can run multiple instances of the Cylance Engine on the same machine (for example, one instance runs the legacy Infinity Daemon Protocol while another instance runs the newer REST API). However, if you have not configured the instances correctly, they may conflict with each other and cause issues. Additionally, one instance may use all the CPU resources, which can starve other instances. BlackBerry does not recommend running multiple instances except for the dual-protocol scenario because a single instance can service a large number of simultaneous requests.

If you are running multiple instances, BlackBerry recommends that each instance have a separate `.ini` configuration file with unique settings for the following:

- **Port:** The TCP port cannot be shared and each instance requires a unique port. If desired, you can specify the port on the command line instead of in the `.ini` configuration file.
- **Max concurrency:** The number of cores available for the Cylance Engine should be divided among the instances so that the machine is not oversubscribed. Letting all instances have access to all CPU cores can

result in timeouts and sluggish responses. If desired, you can specify the max concurrency on the command line instead of the .ini configuration file.

- The file path for file logging: If the log file is not unique, it may not be possible to discern which instance logged a given message.
- The temporary archive directory (if used): Using separate directories ensures that one instance does not accidentally try to clean up after another instance.
- The manifest-based centroid directory (if used): The centroids stored in this directory are periodically updated. During an update, if another instance tries to access any of the files in this directory, that instance could run into errors.

Installing and updating the Cylance Engine

You can install the Cylance Engine on a supported Linux distribution or Windows server.

Install the Cylance Engine on a Linux distribution

1. Download the appropriate Cylance Engine package for your Linux distribution.
 - The installation package for CentOS, Fedora, and Red Hat Enterprise Linux (RHEL) distributions is an rpm package (for example, SampleScore-<version>.rpm).
 - The installation package for Debian and Ubuntu distributions is a deb package (for example, samplescore_<version>_amd64.deb).
 - If you want to install the Cylance Engine without a distribution package manager, the installation package is a zip file (for example, Cylance.Engine-linux-x64-<version>.zip).
2. Do one of the following:

Task	Steps
Install the Cylance Engine on a CentOS, Fedora, or Red Hat distribution.	In the command prompt, run the following command, where <i>package</i> is the complete name of the rpm package. <pre>rpm -ivh package</pre>
Install the Cylance Engine on a Debian or Ubuntu distribution.	In the command prompt, run the following command, where <i>package</i> is the complete name of the rpm package. <pre>dpkg -i package</pre>
Install the Cylance Engine without a distribution package manager.	The only requirement for the location of the files from the package is that the log file path must point to a writable location.

Query the version of your Cylance Engine on a Linux distribution

To determine the currently installed version of the Cylance Engine, open the command prompt and run the following command:

```
cat /opt/cylance/share/doc/samplescore/VERSION
```

This file may not exist in older versions of the Cylance Engine, in which case open the command prompt and run one of the following commands:

- For a CentOS, Fedora, or Red Hat distribution:

```
rpm -qi SampleScore
```

- For a Debian or Ubuntu distribution:

```
dpkg -s samplescore
```

Update the version of your Cylance Engine on a Linux distribution

1. To update an older version of the Cylance Engine to a newer version, download the newer installation package for your distribution.
2. Do one of the following:

Note: You can also update the Cylance Engine by first uninstalling the older version (see [Remove the Cylance Engine from a Linux distribution](#)) and then installing the newer version (see [Install the Cylance Engine on a Linux distribution](#)).

Task	Steps
Update the Cylance Engine on a CentOS, Fedora, or Red Hat distribution.	In the command prompt, run the following command, where <i>new-package</i> is the complete name of the rpm package. <pre>rpm -U new-package</pre>
Update the Cylance Engine on a Debian or Ubuntu distribution.	In the command prompt, run the following command, where <i>new-package</i> is the complete name of the deb package. <pre>dpkg -i new-package</pre>
Update the Cylance Engine without a distribution package manager.	The only requirement for the location of the files from the package is that the log file path must point to a writable location.

Remove the Cylance Engine from a Linux distribution

To uninstall the Cylance Engine from a Linux distribution, do one of the following:

Task	Steps
Uninstall the Cylance Engine from a CentOS, Fedora, or Red Hat distribution.	In the command prompt, run the following command: <pre>rpm -e 'rpm -qa grep -i samplescore'</pre>
Uninstall the Cylance Engine from a Debian or Ubuntu distribution.	In the command prompt, run the following command: <pre>dpkg -r samplescore</pre>

Install the Cylance Engine on a Windows distribution

1. Download the appropriate Cylance Engine package.
 - To install the Cylance Engine with the MSI installer, the installation package is an msi package (for example, Cylance.Engine.Service-<version>.msi).

- To install the Cylance Engine without an installer, the installation package is a zip file (for example, Cylance.Engine-Win-x64-<version>.zip).

2. Do one of the following:

Task	Steps
Install the Cylance Engine with the MSI installer.	Run the MSI installer using an account with sufficient permissions to install and start a system service.
Install the Cylance Engine without an installer.	The only requirement for the location of the files from the package is that the log file path must point to a writable location.

Query the version of your Cylance Engine on a Windows distribution

To determine the currently installed version of the Cylance Engine, go to the Add or Remove programs option in the Windows settings.

Update the version of your Cylance Engine on a Windows distribution

1. To update an older version of the Cylance Engine to a newer version, download the newer installation package.
2. Do one of the following:

Task	Steps
Update the Cylance Engine with the MSI installer.	Run the MSI installer using an account with sufficient privileges to install and start a system service.
Update the Cylance Engine without an installer.	The only requirement for the location of the files from the package is that the log file path must point to a writable location.

Remove the Cylance Engine from a Windows distribution

To uninstall the Cylance Engine, go to the Add or Remove programs option in the Windows settings.

File-scoring service

The Cylance Engine file-scoring service (CylanceTcpService) provides a method to score individual files and archives.

Sample scored service script

The samplescored service script allows the user to start and stop the service using traditional service commands in Linux.

The samplescored service script wraps access to the Cylance Engine, which analyzes a given file or archive and returns a confidence score.

The root user can start and stop the samplescored service script using the following commands:

```
service samplescored start
```

```
service samplescored stop
```

```
service samplescored status
```

By default, the service startup scripts are configured to run the service as root. However, you can change this to any account by editing the `/opt/cylance/bin/start_daemon` script to change the `$SERVICE_USER` to the desired account.

Sentinel file

Each instance of Cylance Engine creates a sentinel file in the temporary directory that you can use to verify whether the service is running, the protocol that is being used, and the TCP port that it is listening on.

The sentinel file is a zero-length file with the configured port as the name of the file. It is created after all initialization is complete and right before the service starts listening for connections. The presence of the file indicates that the service is running and ready for requests.

You can specify the location of this sentinel file with the `CYLANCE_TCP_SERVICE_TEMP_PATH` environment variable. The path of the sentinel follows the pattern of:

```
$TEMP/Cylance/CylanceTcpService/$pid/$protocol/tcp/$port
```

where:

- `$TEMP` is the system default temporary directory or the directory specified by the `CYLANCE_TCP_SERVICE_TEMP_PATH` environment variable.
- `$pid` is the process identifier for the instance of the service.
- `$protocol` is the identifier for the protocol. Currently, this can be `bpv1` for the Infinity Daemon Protocol or `CERAv1` for the REST API.
- `$port` is the configured port on which this instance is waiting.

Environment variables

The Cylance Engine supports reading environment variables that control behavior before the configuration file is loaded.

Variable	Description
CYLANCE_TCP_SERVICE_TEMP_PATH	This specifies the path where the sentinel file for the instance is created. By default, the path is <code>/tmp</code> , but this can be redirected to another path using this variable. The Cylance Engine automatically removes this file when the service shuts down.

Command-line options for the Cylance Engine

The Cylance Engine uses a `.ini` configuration file to control its configuration (see [Configuration file for the Cylance Engine](#)) but you can override certain options with command-line options.

Option	Default setting in <code>.ini</code> configuration file (Service section)	Description
<code>-c</code>	<code>./CylanceTcpService.ini</code>	<p>This option specifies the path to the configuration file. This option is only required when running the Cylance Engine in a different directory from where it was installed.</p> <p>The default location of the <code>CylanceTcpService.ini</code> configuration file is <code>/opt/cylance/lib/samplescore</code>.</p>
<code>-p</code>	<code>Port=9002</code>	<p>This option specifies the port that this instance of the Cylance Engine should listen on for connections. Any port number greater than 1024 will work but the port must not be occupied by another application.</p> <p>If a port number is specified in the Service section of the <code>CylanceTcpService.ini</code> configuration file, this option overrides that value.</p>
<code>-s</code>	<code>Shutdown=false</code>	<p>This option enables the processing of the shutdown command via the socket.</p> <p> CAUTION: This option should not be enabled for production environments.</p>
<code>-u</code>	<code>DataFileUpdateInterval=10</code>	<p>This option specifies the interval, in seconds, between checks to see whether the <code>.ini</code> configuration file contains any updated settings. Only certain settings in the configuration file are checked. For more information, see the Service section in Configuration file for the Cylance Engine.</p>

Option	Default setting in .ini configuration file (Service section)	Description
-C	MaxConcurrency=0	This option specifies the maximum number of files that the Cylance Engine can process simultaneously. The default value of 0 indicates that the concurrency should be set to the number of detected CPU cores. Setting this value higher than the number of CPU cores could lead to poor system performance.
-P	MaxPendingConnections=100	This option specifies the maximum number of connections that can be pending (that is, not yet being processed) before new connections are rejected.
-T	ScoringTimeout=300	This option specifies the amount of time, in seconds, to allow for scoring a single file. A non-zero value indicates the time in seconds. Setting this value too low may result in abort errors when scoring files, especially archives.
-E	All configured activities	This option disables all activities specified in the configuration file except for the list provided in this command. In the .ini configuration file, the activity names are specified by the name that follows "Activity:". Multiple activities can be specified by using a comma-separated list without spaces.
-D	—	This option disables the activity or activities specified by the list provided in this command. In the .ini configuration file, the activity names are specified by the name that follows "Activity:". Multiple activities can be specified by using a comma-separated list without spaces.
--protocol	Protocol=IDP	<p>This option specifies the protocol to run on the listening port. The valid values are:</p> <ul style="list-style-type: none"> InfinityDaemonProtocol or IDP: the legacy protocol supported by TcpShim and InfinityTcpService, and earlier versions of CylanceTcpService. REST: the newer Cylance RESTful API (CERA) protocol <p>When using the Mono runtime, the default protocol is IDP, whereas when running under the Microsoft .NET Framework, the default protocol is REST.</p>
--prometheus-port	—	This option specifies the port on which to serve the Prometheus server.

The Cylance Engine has two additional options that control logging and that map to sections other than the Service section:

- The option `--console-log-level` corresponds to the `LogLevel` setting in the `ConsoleLog` section.

- The option `--file-log-level` corresponds to the `LogLevel` setting in the `FileLog` section.

These options allow you to override temporarily the logging settings to help diagnose issues without having to change the `.ini` configuration file.

All supported command-line options can always be queried using the `-h` or `--help` option.

The following invocation shows a command including options:

```
/opt/cylance/bin/CylanceTcpService -c CylanceTcpService.ini -p 9002 -u 30
```

When a given parameter is found both in the configuration file and on the command line, the command-line option overrides the configuration file. This allows you to start multiple instances of the Cylance Engine on different ports while sharing the same configuration file.

Configuration file for the Cylance Engine

The configuration file is a standard UTF-8 encoded text file that is broken into multiple sections.

- Each section name must be enclosed in square brackets (`[]`).
- Each section contains zero or more key/value pairs with the format `key=value`.
- Whitespace around the `=` sign is ignored, as are blank lines and trailing whitespace.
- You can add comments by starting a line with a `#` sign. The comment extends to the end of that line.

You can update certain entries in the `.ini` configuration file while the Cylance Engine is running. The `DataFileUpdateInterval` setting determines how often the service checks for these changes. For most settings, you must restart the service for the updated setting to take effect. Settings that you can update while the service is running are indicated in their respective sections.

For settings that do not have a default value, a `–` symbol appears in default-value column in the tables.

The `Service` section controls general features and configuration controlling the service. You must restart the service for updates to any of the settings in this section to take effect.

Key	Default value	Description
<code>DataFileUpdateInterval</code>	<code>0</code>	<p>This key specifies the interval, in seconds, between checks to see whether the <code>.ini</code> configuration file contains any updated settings.</p> <ul style="list-style-type: none"> • A value of 0 disables checking whether any settings have been updated. • A value from 1 to 10 indicates the default interval of 10 seconds (that is, values less than 10 are treated as the minimum of 10 seconds). • A value greater than 10 is the interval in seconds.
<code>ExternalClientEnable</code>	<code>false</code>	<p>When this value is false, the service restricts connections to the local host only.</p> <p>When this value is true, the service listens on all network interfaces, allowing clients from other machines to connect to the service.</p>

Key	Default value	Description
IP	127.0.0.1	This key specifies the TCP port the Cylance Engine should listen on when multiple ports are available. Setting the value to 0.0.0.0 listens on all interfaces, which is the same as setting the ExternalClientEnable key to true.
MaxConcurrency	0	This key specifies the maximum number of files that the Cylance Engine can process simultaneously. The default value of 0 indicates that the concurrency should be set to the number of detected CPU cores. Setting this value higher than the number of CPU cores could lead to poor system performance.
MaxPendingConnections	100	This key specifies the maximum number of connections that can be pending (that is, not yet being processed) before new connections are rejected.
Port	–	This key specifies the port on which the TCP Service should listen.
Protocol	IDP for Mono; REST for Microsoft .NET 8.0	This key specifies the protocol to run on the listening port. The valid values are: <ul style="list-style-type: none"> InfinityDaemonProtocol or IDP: the legacy protocol supported by TcpShim, InfinityTcpService, and earlier versions of the CylanceTcpService. REST: the newer Cylance RESTful API (CERA) protocol
Proxy	–	This key specifies a proxy server to use when making any outgoing connections. The format of the proxy specification is “http://proxy.com:port”, or “http://username:password@proxy.com:port” if a username and password are required. The HTTPS protocol can be used as well. When using a username and password, you must escape any special characters used in the URL notation (for example, /, :, @, and so on), according to the HTTP specification.
ScoringTimeout	0	This key specifies the amount of time, in seconds, to allow for scoring a single file. The default (0) indicates 300 seconds (5 minutes). A non-zero value indicates the time in seconds. Setting this value too low may result in abort errors when scoring files, especially archives.
ShutdownCommand	false	This key specifies whether the Cylance Engine should honor the shutdown command (s) or not. The default value of false ignores the shutdown command.

When Protocol=REST, you can configure additional settings in the Service section to control secure connections.

Key	Default value	Description
HttpsEnable	false	When this value is false, the service does not enable HTTPS connections. When this value is true, the service enables HTTPS connections. When they are enabled, you must configure the TlsPort, TlsCertPath, and TlsCertPassword settings.
TlsPort	—	This key specifies the port for secure connections, that is, for clients that connect to the REST API via HTTPS. The Port setting is for insecure connections via HTTP only.
TlsCertPath	—	This key specifies the path to a file containing the TLS certificate in Persona Information Exchange (.pfx) format. This is a standard X.509 certificate that contains both the public and private key.
TlsCertPassword	—	This key specifies the password to use for the PFX TLS certificate.
ClientCertRequired	false	When this value is false, client certificates are not validated. When this value is true, client certificates are validated that they are signed by the certificate specified by TlsCertPath. Clients that cannot present a correctly signed certificate are denied access.
CacheSize	1024	This key specifies the maximum number of validated certificates to cache.
CacheEntryExpiration	1	This key specifies the expiration time, in hours, for validated certificates in the cache. Once this time has elapsed, the client certificate must be revalidated on connection.

The AllowedRestrictedHashList section indicates where the allowed and restricted SHA256 list can be located. The file path is checked for changes according to the DataFileUpdateInterval setting in the Service section.

Key	Default value	Description
FilePath	—	This key specifies the path to a file containing the allowed and restricted hash list JSON entries. For the format of this file, see Infinity Public Data API .

The ConsoleLog section controls the logging of the Cylance Engine to the console. You must restart the service for updates to any of the settings in this section to take effect.

Key	Default value	Description
LogLevel	–	<p>This key specifies the verbosity of the console log. The valid values are</p> <ul style="list-style-type: none"> • None – this value disables console logging • Error • Warning • Info • Debug • Verbose <p>It is not recommended to set the level higher than necessary because it can produce a lot of output, but may be able to help diagnose an issue if there are problems.</p>
LogTag	false	<p>This key specifies whether tagging information should be included with each message. When false, specifies that no tagging information should be included; when true, indicates that the module name space and message hash will be included with each message.</p> <p>The tag represents the particular subsystem that produced the message. For most messages, this information is not important and you can disable the setting to reduce the size of the log file.</p>
TimeStamp	off	<p>This key specifies the format of the time stamp included with each message. The default value of off indicates that no time stamp should be included. Other valid values are:</p> <ul style="list-style-type: none"> • Off, which indicates that no time stamp should be included • Local, which prints the local time in ISO 8601 format • UTC, which prints the UTC time in ISO 8601 format • Simple, which prints a simple, non-ISO 8601 time stamp

The FileLog section controls the logging of the Cylance Engine to a file. The FileLog section supports the same settings as the ConsoleLog section, and the following additional setting. You must restart the service for updates to any of the settings in this section to take effect.

Note: Because the console logging and file logging are separate sections within the configuration file, you can configure the settings for each section with different values.

Key	Default value	Description
FilePath	–	This key specifies the path to the file in which to log the messages. Relative paths are relative to the location of the .ini configuration file.

The Syslog section controls the syslog configuration. You must restart the service for updates to any of the settings in this section to take effect.

Key	Default value	Description
Host	—	When specified, logging to a system logger (syslog) is enabled and this setting specifies the host name of the service. The default is to leave this setting blank and not log to a syslog server.
Port	514	This key specifies the port on which the syslog service is running. For most systems, the default is 514, but you can specify a different port for non-standard configurations.
Protocol	RFC5424	This key specifies the protocol to use when communicating with the syslog service. The default protocol is RFC5424, but for older services, RFC3164 is supported as well.
Facility	Security	<p>This key specifies the syslog facility that you log any messages under. The valid values are:</p> <ul style="list-style-type: none"> • Kernel • User • Mail • System • Security • Internal • Printer • News • UUCP • Clock • Security2 • Ftp • Ntp • Audit • Alert • Clock2 • Local0 • Local1 • Local2 • Local3 • Local4 • Local5

Key	Default value	Description
MalwareSeverity	Alert	This key controls the syslog level at which malware notifications are sent. The valid values are: <ul style="list-style-type: none"> • Emergency • Alert • Critical • Error • Warning • Notice • Informational • Debug
SeverityFilter	Warning	This key controls which messages get logged with the syslog service. Any message with a lower priority is logged. The valid values are: <ul style="list-style-type: none"> • Error • Warning • Info
ScoreThreshold	0.0	This key specifies the threshold at which the score is considered malware. Everything equal to or less than this value results in a syslog malware notification. The range of valid values is +1.0 to -1.0.

The CloudScoring section controls whether scores for files should be retrieved from the Cylance Infinity Cloud or only calculated locally. You must restart the service for updates to any of the settings in this section to take effect.

Key	Default value	Description
Enabled	false	When this value is false, the service disables the retrieval of cloud scores. When this value is true, the service enables the retrieval of cloud scores. If you enable setting, you must also specify an API key via the ApiKey setting.
ApiKey		This is the Cylance Infinity Cloud API key to use for cloud-scoring requests. This must be a 32-character, all upper-case, hexadecimal number. API keys are assigned per customer. Contact your BlackBerry representative if you need an API key
RequestTimeoutMs	–	This key specifies the time, in milliseconds, to wait for a response from the cloud. Normally, responses are very quick (within hundreds of milliseconds) but they can take longer depending on network conditions.

The ManifestCentroidUpdate section controls the retrieval and loading of the new manifest-based centroids, which allows the service to fetch only the centroids that it does not already have. You must restart the service for updates to any of the settings in this section to take effect.

Key	Default value	Description
Enabled	false	When this value is false, the service disables the retrieval of new centroids. When this value is true, the service enables the retrieval of new centroids.
CentroidUpdateIntervalHours	48	This key specifies the interval, in hours, between checks for new centroids.
DirectoryPath	—	This key specifies the directory in which to store the downloaded centroids. The default is to store them under an appropriate local application data folder on the system.

The Cache section controls the internal caching of scores to avoid rescoring commonly seen files. You must restart the service for updates to any of the settings in this section to take effect.

Key	Default value	Description
CacheSizeMB	—	This key specifies the amount of memory, in MB, to use for the cache. When score caching is enabled, the minimum value is 1MB. While there is no upper limit to the size, large caches are not recommended because this memory is exclusively reserved for the cache.
Enabled	false	When this value is false, the service disables score caching. When this value is true, the service enables score caching.
RedisHost	—	This key specifies the URL of the Redis server to use for caching, instead of the in-memory score cache. When this setting is a valid URL (using the format redis://host:port/), the internal cache is disabled and the CacheSizeMB key is ignored. Scores that are stored in the Redis cache still honor the TimeToLive key, which is enforced by the Redis server and not the Cylance Engine. You should configure multiple instances of the Cylance Engine to use the same Redis server to share the score caching among them.

Key	Default value	Description
TimeToLive	–	<p>This key specifies the length of time, in hours, for which any cache entry is kept in the cache. The minimum is 1 hour but this value can be increased to allow entries to live longer. For the local cache (not Redis), cache entries are not automatically expunged when they exceed their time to live, but they may be evicted any time after this period, depending on the need for space in the cache.</p> <p>This setting does not have an inherent maximum value, but we recommend that the TimeToLive value not be set too high when cloud scoring is enabled to get updated values from the cloud.</p>

The activity sections control which activity classes are loaded. Each section must have a unique name and start with "Activity:". Although the name does not matter, it is recommended that scoring activities start with "Score-" and explaining activities start with "Explain-" (for example, "Score-PE" and "Explain-PE"). Apart from the amount of available memory, there is no limit for the number of configured activities. Multiple generations of a single model (for example, PE) can be specified if the activity name is unique, such as Score-PE4 and Score-PE6.

The activities support a number of settings but not all are valid for each type of activity. Note that, in the activity sections, only the Centroids setting is checked for changes according to the DataFileUpdateInterval setting in the Service section.

Key	Description
AssemblyPath	This key applies to scoring, explaining, and archive activities. This is the path, relative to the .ini configuration file, to the assembly for the activity.
EnsemblePath	This key applies to scoring and archive activities only. This is the path, relative to the .ini configuration file, to the ensemble for the activity.
Centroids	<p>This key applies to scoring activities only. This is an optional path to a file containing centroids to load into the model. Centroids do not have to be separated by model; each scoring activity loads only the appropriate centroids from the specified file. Therefore, all centroid settings can point to the same file; this is the recommended setup.</p> <p>In the activity sections, only the Centroids setting is checked for changes according to the DataFileUpdateInterval setting in the Service section.</p>
MaxNestedFileDepth	This key applies to the archive activity only. When the service is processing archives, this value controls how many levels of nesting of archives to examine before giving up. A value of 1 examines the top-level archive entries but no archives inside. Setting this value too high may result in the scoring operation aborting due to a timeout, or the process hanging if the archive is malformed.
Passwords	This key applies to the archive activity only. This is an optional, comma-separated list of default passwords to try on password-protected archives. These are tried after any passwords that are passed via the Score or Explain commands.

Key	Description
TempArchiveDirectory	This key applies to the archive activity only. When processing archives, if this setting is present, the value is a temporary disk location in which you can extract the contents of the archive for processing. By default, archives are processed entirely in memory. Using this setting can reduce the amount of memory used for archive processing at the expense of additional disk space usage and processing time.

The following is an example of each activity:

```

...

[Activity:Score-PE]
AssemblyPath=Cylance.Model.SS3PE6.dll
EnsemblePath=Ensemble-20180813-S3V8-PE6.cym
Centroids=Centroids.cyb

[Activity:Explain-PE]
AssemblyPath=SampleExplainPE.dll

[Activity:Score-ARC]
AssemblyPath=Cylance.Model.ARC.dll
EnsemblePath=Ensemble-20190319-S0V5-ARC.cym
MaxNestedFileDepth=3
Passwords=abc123,clean,dirty

...

```

The Activity:Score-ARC name is not special. The Cylance.Model.ARC.dll assembly defines this as being an archive activity. Only Cylance.Model.ARC.dll understands the MaxNestedFileDepth and Passwords settings. If these settings are present in any other section, the service ignores them.

Note: The Archive section is no longer supported and has been replaced with Activity:Score-ARC.

The Prometheus section allows you to enable whether you want to use a Prometheus server to scrape metrics data from the Cylance Engine periodically, and the port to listen on for scrape requests. For more information about how Prometheus works with the Cylance Engine, see [Appendix: Prometheus monitoring support](#).

Key	Default	Description
Enabled	false	This key specifies whether to use a Prometheus server to scrape metrics data. When enabled, the service listens on the TCP port that you specify using the Port key, or the port specified using the <code>--prometheus-port</code> command-line option (see Command-line options for the Cylance Engine).
Port	9009	This key specifies the port to listen on for scrape requests.

File-scoring service protocols

The Cylance Engine file-scoring service supports three protocols:

- A REST-based API called the Cylance Engine RESTful API (CERA), based on standard HTTP and HTTPS connections using JSON for both requests and responses. This protocol was introduced in Cylance Engine v0.11. Note that CERA is not available in Mono-based packages.
- An Internet Content Adaptation Protocol (ICAP) service, based on the ICAP protocol ([rfc3507](#)). The service is situated between an ICAP client and the file-scoring service using the CERA. For documentation, see the ICAP service package that is downloaded separately.
- The legacy Infinity Daemon Protocol (IDP), also known as the CylanceTcpService Protocol, is the proprietary binary protocol supported by TcpShim, InfinityTcpService, and CylanceTcpService.

To use the service with this protocol, a TCP connection must be established with the server. The service can be configured to listen on any valid TCP port number (1024 - 65535). Port 9002 is the default port in InfinityDaemonClient, samplescored, samplescore, and ttstatic. The port used by the service can be customized using the configuration file or the `p` or `--port` command-line option.

For more information, see [Appendix: CylanceTcpService Protocol](#).

The protocols serve as a bridge between client code and Cylance Engine activities, acting as a generic service, providing the infrastructure to process files and hand the results back to the client.

Cylance RESTful API

The Cylance RESTful API (CERA) protocol uses standard HTTP or HTTPS for sending commands and receiving the responses. The [curl utility](#) can be used to score or retrieve threat indicators for files. In addition, `cera-client` is an example CERA client that uses the [Python requests library](#). `cera-client` provides a convenient command-line interface for scoring or explaining directories of files with a controllable degree of parallelism.

The default protocol is the Infinity Daemon Protocol. To use the CERA protocol, you must enable the Infinity Daemon Protocol in the `CylanceTcpService.ini` configuration file or via a command-line option. On the command line, the `--protocol` option specifies which protocol to use for this invocation only. The valid values are:

- 'InfinityDaemonClient' or 'IDP': enables the default Infinity Daemon Protocol.
- 'CylanceEngineRestApi', 'CERA', or 'REST': enables the new RESTful API (Microsoft .NET 5.0 or later only).

The entries are not case-sensitive (for example, 'CERA', 'cera', and 'Cera' are all treated the same).

Similarly, the protocol can be specified in the `CylanceTcpService.ini` configuration file:

```
[Service]
Protocol=REST
```

This example makes the REST protocol the default for all invocations unless it is overridden via the `--protocol` command-line option. The options for this setting are the same as the options described above for the `--protocol` command-line option.

The port setting in the `.ini` configuration file and the `--port` option specify the port that the Cylance Engine lists for incoming REST connections via HTTP only. Another option allows for HTTPS connections when the service is properly configured. In the examples below, `curl` is being run on the same machine with the default TCP port (9002) so all connections are to `localhost:9002` or `127.0.0.1:9002`.

Getting model details

To retrieve information about the currently loaded models, in the command prompt, run the following command:

```
GET /apiv1/models/
```

The response is a JSON object with the details for all loaded models. The exact list of models may be different depending on how the service is configured and the exact version of the Cylance Engine.

The following example uses curl with the Cylance Engine running on the default port 9002.

Note: For brevity, some curl output has been removed and a JSON pretty-printer has been applied to make the JSON easier to read. By default, curl prints the JSON raw, which is difficult to read.

```
$ curl http://localhost:9002/apiv1/models/
{
  "Status": "OK",
  "Models":
  [
    {
      "Banner": "...",
      "ModelVersion": 131975059429967678,
      "SampleFormat": "ARC",
      "Generation": 1,
      "SubGeneration": "A",
      "EnsembleVersion": 5,
      "CentroidHash": 0
    },
    {
      "Banner": "...",
      "ModelVersion": "131774598312539083",
      "SampleFormat": "ELF",
      "Generation": 2,
      "SubGeneration": "A",
      "EnsembleVersion": 6,
      "CentroidHash": "3979322740683475419"
    },
    {
      "Banner": "...",
      "ModelVersion": 132713821343850774,
      "SampleFormat": "MO",
      "Generation": 3,
      "SubGeneration": "A",
      "EnsembleVersion": 2,
      "CentroidHash": "7085422129475707133"
    },
    {
      "Banner": "...",
      "ModelVersion": "132624646650794342",
      "SampleFormat": "MOFAT",
      "Generation": 1,
      "SubGeneration": "A",
      "EnsembleVersion": 1,
      "CentroidHash": 0
    },
    {
      "Banner": "...",
      "ModelVersion": "131764252418431270",
      "SampleFormat": "OLE",
      "Generation": 3,
```

```

    "SubGeneration": "A",
    "EnsembleVersion": 2,
    "CentroidHash": 0
  },
  {
    "Banner": "...",
    "ModelVersion": "131764272845001270",
    "SampleFormat": "OOXML",
    "Generation": 3,
    "SubGeneration": "A",
    "EnsembleVersion": 3,
    "CentroidHash": 0
  },
  {
    "Banner": "...",
    "ModelVersion": "132253137167261698",
    "SampleFormat": "PDF",
    "Generation": 3,
    "SubGeneration": "A",
    "EnsembleVersion": 3,
    "CentroidHash": 0
  },
  {
    "Banner": "...",
    "ModelVersion": "131786662583688997",
    "SampleFormat": "PE",
    "Generation": 6,
    "SubGeneration": "A",
    "EnsembleVersion": 9,
    "CentroidHash": "1359238976895146529"
  }
]
}

```

In this example, the Banner lines have been truncated because they return long strings. Each object has the following keys:

Key	Type	Description
Banner	string	This is the banner string from the ensemble containing the model release date and copyright information.
ModelVersion	string	This is the version number of this model. This number can be used to access centroids for the model.
SampleFormat	string	This is the format of the file accepted by the model (for example, "PE" or "MO").
Generation	number	This is the generation number of the model.
SubGeneration	string	This is the sub-generation of the model.
EnsembleVersion	number	This is the version of the ensemble. The version is incremented when new content is added (for example, when the default set of centroids is updated).

Key	Type	Description
CentroidHash	string	This is a 64-bit hash identifier for the set of centroids loaded into the model. A value of 0 means it has no centroids. Because this is a hash value, it can only be used to see if the set is different, but comparing two centroids' hashes does not indicate which set is newer.

You can retrieve information on a specific model by including the model version. In this example, only one model is returned.

```
GET /apiv1/models?ver=132253137167261698
```

You can also retrieve information for multiple models. In this second example, because you are requesting two models, the resulting JSON contains exactly two JSON objects.

```
GET /apiv1/models?ver=132253137167261698&ver=131786662583688997
```

Scoring a file

To score a file, the REST API uses the HTTP PUT method; the POST method is not supported. The file can be provided by:

- A file path: The full path to the file that is visible to the Cylance Engine. The file path is passed via a small block of JSON. Note that this is the only way to specify a file to score using the Infinity Daemon Protocol.
- A binary: Since the PUT method allows transfer of binary data, the file can be directly transferred to the TCP Service via the HTTP(S) connection. Because the Cylance Engine supports multiple compression formats, files can be compressed in order to reduce transmission bandwidth.

When scoring by file path, the client posts a JSON object with the file path to the Cylance Engine:

```
PUT /apiv1/score
Content-Type: application/json
Content-Length: 36

{
  "FilePath": "/tmp/infinityd.exe"
}
```

In Cylance Engine 1.2 and later, you can score multiple files in a single request using both JSON and MIME multipart:

```
PUT /apiv1/score
Content-Type: application/json
Content-Length: 70

{
  "FilePaths": [ "C:/tmp/sample1.exe", "C:/tmp/sample2.exe" ]
}
```

If the body is not valid JSON, a 400 Bad Request error is returned. For a 200 OK response, a block of JSON is returned that represents the scoring result. When scoring multiple files, an array of results are returned.

```
[
  {
```

```

    "Status": "OK",
    "SamplePath": "C:/tmp/sample1.exe",
    "AggregateScore": 1.0,
    "Sha256":
"5AE1246EAADE01C5840338850D7B35BF70243FC13A8E006642445DB08CB42A50",
    "MaxDepthExceeded": false,
    "SampleFormatUnknown": false,
    "Scores": [
      {
        "Score": 1.0,
        "Determinant": "MODEL",
        "SampleFormat": "PE",
        "ModelVersion": 133368370684345704,
        "Source": "LOCAL_ENDPOINT",
        "Classifier": "ML",
        "ParseStatus": "OK",
        "CentroidHash": "0"
      }
    ]
  },
  {
    "Status": "OK",
    "SamplePath": "C:/tmp/sample2.exe",
    "AggregateScore": 1.0,
    "Sha256":
"7E668791A2089E7CB82B09D4574B63DCE1B13DA7278E5F98075F4A070C09AB6D",
    "MaxDepthExceeded": false,
    "SampleFormatUnknown": false,
    "Scores": [
      {
        "Score": 1.0,
        "Determinant": "MODEL",
        "SampleFormat": "PE",
        "ModelVersion": 133368370684345704,
        "Source": "LOCAL_ENDPOINT",
        "Classifier": "ML",
        "ParseStatus": "OK",
        "CentroidHash": "0"
      }
    ]
  }
]

```

The keys of the scoring-result object are:

Key	Description
Status	<p>This indicates the status of the operation.</p> <ul style="list-style-type: none"> • If the file scored successfully, OK is returned. • If an error occurs, a message describing the error is returned. <p>This field is present only in top-level scoring-result objects; it is omitted for any objects in the Children field.</p>

Key	Description
AggregateScore	<p>If the file produced a single result, this is the aggregate score. If the file produced multiple results (for example if the file is an archive), the aggregate score is the lowest score produced among all of the files.</p> <p>This field may be interpreted as the overall score for the file but the individual scores are also available for inspection. Non-scores such as 'NaN' are filtered out before finding the lowest score so that nested files that could not be processed do not corrupt valid results.</p>
MaxDepthExceeded	<p>When scoring an archive, a value of true indicates that the entire archive was not explored because the nesting level was higher than the configured maximum. This means that only a partial result has been returned. A value of false indicates that the entire archive has been examined and all results are available.</p>
SampleFormatUnknown	<p>When the value for this key is true, the file could not be scored by any models that are currently loaded. This normally occurs because the file is not of a supported type.</p>
Scores	<p>This is the list of one or more scores for the file. In most cases, there is only one score unless the file was scored by multiple models.</p>
Children	<p>This is the list of child scoring results for this file. This key is normally seen only when scoring an archive or a MOFAT file. When present, it contains the list of scoring-result objects for any files inside the archive. If one of those file is also an archive, it too has child scoring results until the configured maximum nested depth is reached.</p>
SamplePath	<p>This is the file's path.</p> <ul style="list-style-type: none"> • When scoring by filepath, this is equivalent to the request's FilePath. • When scoring using binary data, this is equal to the SHA256 of the file. <p>For child scoring results (for example, in the case of files within an archive), the child's relative file path (that is, its path within an archive) is prefixed with a pipe and appended to the parent SamplePath (for example: archive.zip path/to/sample.exe).</p>

Each scoring result contains the following information:

Key	Description
Score	<p>This is the score for the file.</p> <ul style="list-style-type: none"> • If an error is generated (the value in the Determinant field is PARSER, CONFIG, or ABORT), this field contains NaN (not a number). • If the value in the Determinant field is WHITECENTROID or WHITELIST, this field is always +1.0. • If the value in the Determinant field is BLACKCENTROID or BLACKLIST, this field is always -1.0.

Key	Description
Determinant	<p>This specifies where the results were obtained from.</p> <ul style="list-style-type: none"> • MODEL indicates that the score was calculated by the machine-learning model. • BLACKCENTROID and WHITECENTROID indicate that a centroid was hit and the score was changed accordingly. • BLACKLIST and WHITELIST indicate that the file hash was explicitly disallowed (that is, appeared in the restricted list) or allowed (that is, appeared in the approved list). • PARSER specifies an error while parsing the file. • ABORT means that the file was aborted before processing was completed because the scoring took longer than the timeout period to complete. • CONFIG indicates that the maximum nested depth was exceeded when processing an archive based on a setting from the CylanceTcpService.ini configuration file.
SampleFormat	This is the type of file that was scored.
ModelVersion	This is the version of the model that produced the score. Because JSON does not handle 64-bit integers well, the version is returned as a string.
Source	<p>This specifies the source of the score. The values can be:</p> <ul style="list-style-type: none"> • LOCAL_ENDPOINT if the score was calculated locally • INFINITY_CLOUD if the score came from the Infinity Cloud Service
ParseStatus	This is the status of the parsing of the file. An OK status indicates that the file was parsed successfully. If the status is not OK, two additional fields, StatusCause and CauseMessage, are included to provide more information about why the file could not be parsed.
Classifier	<p>This further classifies where the score was obtained from. The most common value is ML, indicating that the machine-learning model calculated the score. Other valid values:</p> <ul style="list-style-type: none"> • INFINITY_GENERALSCORE indicates that the score came from a scoring operation in the Infinity Cloud Service. • HUMAN indicates that the score came from human analysis of the file. • INDUSTRY indicates that the score came from a third-party source.

When using a binary rather than a path, the request looks like:

```
PUT /apiv1/score
Content-Type: application/octet-stream
Content-Length: <length>

<binary data>
```

In this case, the Content-Type can be any type except for `application/json`, which indicates that a file path has been provided. The most generic is `application/octet-stream`, but others such as `application/gzip` can also be used.

For scoring multiple files as binary, the Content-Type is multipart/form-data; boundary=something. Both binary and JSON requests can be made in the multipart form, but each section must have the correct Content-Type (application/json or application/octet-stream). Nested multipart forms are not supported.

The output of an archive is hierarchical:

```
$ curl -X PUT -T test.tar http://localhost:9002/apiv1/score
{
  "Status": "OK",
  "SamplePath":
  "5D6D21AB0283E17643B64E856D07ACFEBD6FC52EB4B50AFD3CE6891A2A36ECBE",
  "AggregateScore": 1.0,
  "Sha256": "5D6D21AB0283E17643B64E856D07ACFEBD6FC52EB4B50AFD3CE6891A2A36ECBE",
  "MaxDepthExceeded": false,
  "SampleFormatUnknown": false,
  "Scores": [
    {
      "Score": 1.0,
      "Determinant": "MODEL",
      "SampleFormat": "ARC",
      "ModelVersion": "131975059429967680",
      "Source": "LOCAL_ENDPOINT",
      "Classifier": "ML",
      "ParseStatus": "OK"
      "CentroidHash": "0"
    }
  ],
  "Children": [
    {
      "SamplePath":
      "5D6D21AB0283E17643B64E856D07ACFEBD6FC52EB4B50AFD3CE6891A2A36ECBE|
CommonUtils.dll",
      "AggregateScore": 1.0,
      "Sha256":
      "7F3FD0F31FA0C6C840D917567670DA3B4A01EF7D64826E7326DEE8B32454296D",
      "MaxDepthExceeded": false,
      "SampleFormatUnknown": false,
      "Scores": [
        {
          "Score": 1.0,
          "Determinant": "MODEL",
          "SampleFormat": "PE",
          "ModelVersion": "131786662583689000",
          "Source": "LOCAL_ENDPOINT",
          "Classifier": "ML",
          "ParseStatus": "OK"
          "CentroidHash": "1359238976895146529"
        }
      ]
    }
  ],
  {
    "SamplePath":
    "5D6D21AB0283E17643B64E856D07ACFEBD6FC52EB4B50AFD3CE6891A2A36ECBE|infinityd.exe",
    "AggregateScore": 1.0,
    "Sha256":
    "5AE1246EAAD01C5840338850D7B35BF70243FC13A8E006642445DB08CB42A50",
    "MaxDepthExceeded": false,
    "SampleFormatUnknown": false,
    "Scores": [
```

```

    {
      "Score": 1.0,
      "Determinant": "MODEL",
      "SampleFormat": "PE",
      "ModelVersion": "131786662583689000",
      "Source": "LOCAL_ENDPOINT",
      "Classifier": "ML",
      "ParseStatus": "OK"
      "CentroidHash": "1359238976895146529"
    }
  ]
},
{
  "SamplePath":
  "5D6D21AB0283E17643B64E856D07ACFEBD6FC52EB4B50AFD3CE6891A2A36ECBE |
InfinityDotNet.dll",
  "AggregateScore": 1.0,
  "Sha256":
  "F0A7274835C6D32064ED1D1F09104E881F17ACF544A1ECDF2C430D30D9781EA4",
  "MaxDepthExceeded": false,
  "SampleFormatUnknown": false,
  "Scores": [
    {
      "Score": 1.0,
      "Determinant": "MODEL",
      "SampleFormat": "PE",
      "ModelVersion": "131786662583689000",
      "Source": "LOCAL_ENDPOINT",
      "Classifier": "ML",
      "ParseStatus": "OK"
      "CentroidHash": "1359238976895146529"
    }
  ]
},
{
  "SamplePath":
  "5D6D21AB0283E17643B64E856D07ACFEBD6FC52EB4B50AFD3CE6891A2A36ECBE |
infinitydt.exe",
  "AggregateScore": 1.0,
  "Sha256":
  "19F30312D933256BD983DFC6F120F0521D7C97EFB62CB31C5C286F12E4F3C801",
  "MaxDepthExceeded": false,
  "SampleFormatUnknown": false,
  "Scores": [
    {
      "Score": 1.0,
      "Determinant": "MODEL",
      "SampleFormat": "PE",
      "ModelVersion": "131786662583689000",
      "Source": "LOCAL_ENDPOINT",
      "Classifier": "ML",
      "ParseStatus": "OK"
      "CentroidHash": "1359238976895146529"
    }
  ]
},
{
  "SamplePath":
  "5D6D21AB0283E17643B64E856D07ACFEBD6FC52EB4B50AFD3CE6891A2A36ECBE |
InstallerIDCore.dll",
  "AggregateScore": 1.0,

```

```

    "Sha256":
    "40A3BD9E62336C60DAB2F43E81B8F708882D799D7FAE96746B047B036A3F47F1",
    "MaxDepthExceeded": false,
    "SampleFormatUnknown": false,
    "Scores": [
      {
        "Score": 1.0,
        "Determinant": "MODEL",
        "SampleFormat": "PE",
        "ModelVersion": "131786662583689000",
        "Source": "LOCAL_ENDPOINT",
        "Classifier": "ML",
        "ParseStatus": "OK"
        "CentroidHash": "1359238976895146529"
      }
    ]
  }
}

```

The archive test.tar contains five PE files. The top-level aggregate score is the overall score for the archive. In this example, all the files in the archive are benign, but if they were not, the aggregate score would be the lowest of the all the children.

Each file in the hierarchy gets an aggregate score and a list of individual scores. The reason for this is that any file may be an archive that itself contains multiple files. At each level of the hierarchy, the aggregate score gives a quick overview of that entire file tree.

Explaining the score for a file

To explain the score for a file, the REST API uses the HTTP PUT method. Like scoring, the file can be provided by a file path or binary. When explaining by file path, the client posts a JSON object with the file path to the TCP service:

```

PUT /apiv1/explain
Content-Type: application/json
Content-Length: 36

{
  "FilePath": "/tmp/sample.exe"
}

```

In Cylance Engine 1.2 and later, you can score multiple files in a single request:

```

PUT /apiv1/explain
Content-Type: application/json
Content-Length: 70

{
  "FilePaths": [ "C:/tmp/sample1.exe", "C:/tmp/sample2.exe" ]
}

```

When using a binary rather than a path, the request looks like:

```

PUT /apiv1/explain
Content-Type: application/octet-stream
Content-Length: <length>

```

```
<binary data>
```

A block of JSON is returned that represents the explaining result. When scoring multiple files, an array of results are returned.

```
curl -X PUT -T PEParse.dll http://localhost:9002/apiv1/explain
{
  "Status": "OK",
  "Explain": [
    {
      "TTM": {
        "features": {
          "Deception": {
            "ServiceDLL": true,
            "UsesCompression": true
          },
          "Misc": {
            "PrivEscalationCryptBase": true
          }
        },
        "scores": {
          "Destruction": 0,
          "Deception": 8,
          "Collection": 0,
          "DataLoss": 0,
          "Anomalies": 0,
          "Misc": 20,
          "Extended": 0
        }
      },
      "SampleFormat": "PE",
      "SamplePath":
      "D476484BD9E26928DCC740CCEB4B82C95FB5098BEDED638EF12F692BC8EE945E"
    }
  ]
}
```

The output of an explain operation is similar to that returned by the [InfinityDaemonClient utility](#) but the output differs in a few ways:

- The entire output is valid JSON. The Infinity Daemon Client returns pseudo-JSON with some additional, non-JSON elements.
- The Status field indicates the status of the operation. Normally this is OK, meaning the operation completed successfully. If the field displays any other value, it is a message with an indication of what went wrong with the request.
- The Explain field contains the JSON for the explanation in a format similar to the Infinity Daemon Client.
- The Explain field contains an additional SamplePath element. If the path of the file is known, it is returned in this field. If the path is not known, this field contains the SHA256 hash of the item. In the example above, because the file was submitted as part of the explain request, the original file name was not known and therefore the hash is provided.

Shutting down the service

By default, the Cylance Engine does not allow shutdown requests. If the Cylance Engine is configured to allow shutdown requests, the client may request that it be shut down:

```
PUT /apiv1/shutdown
```

```
Content-Length: 0
```

You do not need to include any data with the request; if any data is supplied, it is ignored. However, the `Content-Length` header must be present or the server rejects the request.

- If shutdown is not allowed (`Shutdown=false` in the configuration file), a 401 Unauthorized result is returned.
- If shutdown is allowed (`Shutdown=true` in the configuration file or if the service was started with the `--shutdown` option), a 200 OK is returned with no data.

Password-protected archives

The Cylance Engine supports the most popular archive formats via the score and explain APIs. For archives that require a password, the password must be passed to the Cylance Engine for extraction.

You can specify passwords in the activity sections of the [Configuration file for the Cylance Engine](#) or pass them as part of the API URI. When passing via the URI, the syntax is the same as described in [Passwords specified for archives](#):

```
PUT /apiv1/score?pw=foo
Content-Type: application/octet-stream
Content-Lenth: <length>

<binary data>
```

If the archive cannot be opened with the given passwords or with the passwords in the configuration file, an error is returned.

Appendix: Cylance Infinity Data Service

The Cylance Infinity service provides a set of RESTful APIs to download centroids or the restricted and allowed list of file hashes to apply to the Cylance Engine. For more information, see [Use of centroids in the Cylance Engine](#) and [Restricted and allowed list of file hashes](#).

The base URL of the Infinity Public Data API is `https://inf-data.cylance.com/apiv2/`.

Authentication of requests

All requests require authentication in the form of an `X-IAUTH` header in all HTTP requests. This is an API key granted via license. If you have any questions about the API key, contact BlackBerry customer support.

Format of the API key code

The API key code must be a 32-character hexadecimal (0-9, A-F) key, with all letters in uppercase. Using lowercase letters results in the request being denied with a 401 (Unauthorized) response.

Example

```
GET /apiv2/centroids HTTP/1.0
Host: inf-data.cylance.com
X-IAUTH: 1234567890ABCDEF1234567890ABCDEF
```

Response status codes

Each API request receives a response with a JSON payload and a standard HTTP status code. In any case other than a status code 200 (OK), the JSON payload provides additional detail regarding the error.

Code	Description
200 (OK)	This code indicates a successful call and operation. The response payload is JSON, structured according to the nature of the request.
204 (No content)	This code indicates a successful call and operation. There are no centroids for the model specified with the <code>model=<num></code> . Models with zero centroids and unknown models both return this result.
400 (Bad request)	This code indicates that there was a problem with the structure of the request or its payload. If the failure can be determined, the response payload identifies the failure in the request. A common cause of this type of error is malformed JSON in the request body or an incorrect URL parameter.
401 (Unauthorized)	This code indicates that there was a problem with the authentication. Either the request does not provide an authentication key, or the authentication key is malformed or not found.

Code	Description
404 (Not found)	This code indicates that a request was made for a resource that does not exist. This most commonly occurs with an improperly formed request URL or an invalid API key.
500 (Internal server error)	This is a catch-all code for any unhandled errors that have occurred on the server. If you encounter this error code, contact BlackBerry customer support.
501 (Not implemented)	This code indicates that a request was made against a resource with an operation that has not been implemented yet. Such operations should be identified accordingly in this documentation.
503 (Service unavailable)	This code indicates that the server is having issues completing the requests. The client should try again later.

Service endpoints

The following service endpoints are supported by the API.

Centroids endpoint

The centroids endpoint allows clients to get the latest restricted and allowed centroids available.

Request

Item	Description
URL	GET /centroids
Parameters	<p><code>format</code> (string, optional): The content format can only be JSON or proto. The default is proto.</p> <p><code>model</code> (int, optional): The specific model of the centroid. The value must be a valid model ID. If a value is not given, the result is centroids for all of the models. If the model has no centroids or the model ID is invalid, a 204 (No Content) response is returned.</p>
Examples	<pre>GET /centroids?format=proto&model=131308214743030001</pre> <pre>GET /centroids?format=json&model=131308214743030001</pre>
Headers	<p>For <code>format=json</code>, it is recommended that you pass the <code>Accept-Encoding: gzip</code> header to compress the centroids.</p> <p>For <code>format=proto</code>, the centroids are already compressed using LZMA compression; specifying the <code>Accept-Encoding: gzip</code> header does not further compress the content.</p>

Response

A successful response returns status code 200 (OK), 204 (No Content), 302 (Moved), or 304 (Not Modified).

- For a status code 200, the response includes the requested centroid document. If the client requests a gzip encoding, the contents are gzipped.
- For a status code 204, it means that there are no centroids for the model specified in the request.
- For a status code 302, in the response header, the location field contains the URL that the client is being redirected to and where the centroids document is found. If the client requests a gzip encoding, the centroid document is gzipped. The URL expiration time is between 1 hour and 1 day.
- For a status code 304, which means that there are no new centroids yet, the client gets an empty response body.

If an error occurred, the service returns either 400 or 503. For either of these response codes, the body will contain the following JSON documentation with a message field describing the error.

Centroids endpoint response

```
{
  "message" : "reason of failure"
}
```

For `format=proto`, the result is a binary file, which can be loaded directly into the scoring models.

For `format=json`, the result has the following structure:

JSON data structure

```
{
  "131037481645565647": {
    "white": {
      "timestamp": 1470700127
      "items": [
        {
          "Identifier": "trusted",
          "Radius": 1.0,
          "DistanceMean": 1.0,
          "DistanceStdDev": 0.2,
          "Kind": "subspace",
          "Type": "PE",
          "Indices": [832, 1562, 4981, 36992],
          "Values": [1.0, 1.0, 1.0, 1.0],
          "Status": "ACTIVE",
          "Timestamp": 1470700127
        },
        {
          "Identifier": "something-deleted",
          "Status": "DELETED",
          "Timestamp": 1470690127
        },
        ...
      ]
    },
    "black": {
      "timestamp": 1470700127
      "items": [
        {
          "Identifier": "mimikatz",
          "Radius": 45.0,
          "DistanceMean": 32.0,
          "DistanceStdDev": 8.0,
          "Kind": "fullspace",
          "Type": "PE",
          "Indices": [832, 834, 1562, 2630],

```



```
}
```

Wblist endpoint

The wblist endpoint allows clients to get the latest restricted and allowed list of file hashes available.

Request

Item	Description
URL	GET /wblist
Parameters	after (int, optional): Unix time stamp truncated to the second.
Examples	GET /wblist?after=1470700100
Headers	It is recommended that you pass the <code>Accept-Encoding: gzip</code> header in order to compress the result.

Response

A successful response returns status code 302. In the response header, the location field contains the URL the client is being redirected to and where the document containing the restricted and allowed list of file hashes is found. If the client requests a compressed, gzip version, the document is gzipped. The URL expires in 1 day.

If an error occurred, the service returns either 400 or 503. For any of these response codes, the body contains the following JSON documentation with a message field describing the error.

Allowed/Restricted endpoint response

```
{
  "message" : "reason of failure"
}
```

The JSON file has the following structure.

JSON data structure

```
{
  "white": {
    "items": [
      <sha256>,
      <sha256>,
      ...
    ]
  },
  "black": {
    "items": [
      <sha256>,
      <sha256>,
      ...
    ]
  }
}
```

Appendix: Threat indicators

Each category represents an area that has been frequently seen in malicious software.

Anomalies

These indicators represent situations where the file has elements that are inconsistent or anomalous in some way. Frequently, these are inconsistencies in structural elements in the file.

Indicator	Description
16bitSubsystem	The file utilizes the 16-bit subsystem. Malware uses this to exist in a less secure and monitored part of the operating system, and frequently to perform privilege escalation attacks.
Anachronism	This PE appears to be lying about when it was written, which is atypical for professionally written software.
AppendedData	This PE has some extra content appended to it, beyond the normal areas of the file. Appended data can frequently be used to embed malicious code or data, and is frequently overlooked by protection systems.
AutoitDbgPrivilege	The Autoit script can perform debug activities.
AutoitManyDllCalls	The Autoit script uses many external DLL calls. The Autoit runtime already has many common functions, therefore using additional functionality from external libraries may be a sign of maliciousness.
AutoitMutex	The Autoit script creates synchronization objects. This is often used by malware to prevent multiple infections of the same target.
AutoitProcessCarving	The Autoit script is likely performing process carving to run its own code that appears to come from another process. This is often done to hinder detection.
AutoitProcessInjection	The Autoit script is likely performing process injection to run code in other processes' context possibly to stay undetected or to steal data.
AutoitRegWrite	The Autoit script writes into Windows registry.
Base64Alphabet	The file contains evidence of usage of Base64 encoding of an alphabet. Malware does this to attempt to avoid common detection or to attack other programs using Base64 encoding.
CommandlineArgsImport	The file imports functions that can be used to read arguments from a command line. Malware uses this to collect information on subsequent runs.
ComplexMultipleFilters	The file contains multiple streams with multiple filters.
ComplexObfuscated-Encoding	The file contains an anomalously high number of obfuscated names.

Indicator	Description
ComplexUnsupportedVersionEmbeddedFiles	The file uses the EmbeddedFiles features from newer versions of the PDF standard than the file declares.
ComplexUnsupportedVersionFlate	The file uses the FlateDecode feature from newer versions of the PDF standard than the file declares.
ComplexUnsupportedVersionJbig2	The file uses the JBIG2Decode feature from newer versions of the PDF standard than the file declares.
ComplexUnsupportedVersionJs	The file uses JavaScript features from newer versions of the PDF standard than the file declares.
ComplexUnsupportedVersionXFA	The file uses XFA features from newer versions of the PDF standard than the file declares.
ComplexUnsupportedVersionXObject	The file uses XObject features from newer versions of the PDF standard than the file declares.
ContainsFlash	The file contains flash objects.
ContainsPE	The file contains embedded executable files.
ContainsU3D	The file contains U3D objects.
InvalidCodePageUsed	The file uses an invalid or unrecognized locale, possibly to avoid detection.
InvalidData	The file metadata is obviously bogus or corrupt.
InvalidStructure	The file structure is not valid. The sizes, metadata, or internal sector allocation table is wrong, which may indicate an exploit.
ManifestMismatch	The file demonstrates an inconsistency in its manifest. Malware does this to avoid detection, but rarely covers its tracks deeply.
NontrivialDLLEP	This PE is a DLL with a nontrivial entry point. This is common among DLLs, but a malicious DLL may use its entry point to take up residence in a process.
NullValuesInStrings	Some strings within the file contain null characters in the middle.
PDFParserArraysContainsNullCount	The file contains an anomalously high number of null values in arrays.
PDFParserArraysHeterogeneousCount	The file contains an anomalously high number of arrays containing different types of elements.
PDFParserMailtoURICount	The file contains an anomalously high number of email links (mailto:).
PDFParserMinPageCount	The file has an unusual structure of page objects, such as a high number of child-page objects per node.

Indicator	Description
PDFParserNamesPound-NameMaxLength	The file may attempt to obfuscate its contents by using long encoded strings.
PDFParserNamesPound-NameMinLength	The file contains an anomalously high minimum length of an escaped name.
PDFParserNamesPound-NameTotalLength	The file may attempt to obfuscate its contents by storing much of its content in encoded strings.
PDFParserNamesPound-NameUpperCount	The file contains an anomalously high number of names escaped with uppercase hexadecimal characters.
PDFParserNamesPound-NameValidCount	The file contains an anomalously high number of valid escaped names.
PDFParserNamesPound-PerNameMaxCount	The file contains an anomalously high maximum number of escaped characters per single name.
PDFParserNamesPound-UnnecessaryCount	The file contains an anomalously high number of unnecessarily escaped names.
PDFParserNumbersLeadingDigitTallies8	The file contains an anomalously high number of numbers that start with 8 in decimal representation.
PDFParserNumbersPlus-Count	The file contains an anomalously high number of numbers with an explicit plus sign.
PDFParserNumbersReal-MaxRawLength	The file contains an anomalously high maximum length of a real number.
PDFParserPageCounts	The file contains an anomalously high number of child-page objects.
PDFParserPageObject-Count	The file contains an anomalously high number of page objects.
PDFParserSizeEOF	The file contains an anomalously long end-of-file sequence(s).
PDFParserStringsHex-LowerCount	The file contains an anomalously high number of strings escaped with lowercase hexadecimal digits.
PDFParserStringsLiteral-StringMaxLength	The file contains an anomalously high maximum length of a literal string.
PDFParserStringsOctal-ZeroPaddedCount	The file contains an anomalously high number of octal escaped characters in strings that are unnecessarily zero-padded.
PDFParserTrailerSpread	The file contains an anomalously large spread between trailer objects.
PDFParserWhitespace-CommentMaxLength	The file contains an anomalously high maximum length for a comment.

Indicator	Description
PDFParserWhitespace-CommentMinLength	The file contains unusual short comments that are not used by reader software.
PDFParserWhitespace-CommentTotalLength	The file contains an unusually large amount of commented-out data.
PDFParserWhitespace-EOL0ACount	The file contains an anomalously high number of short end-of-line characters.
PDFParserWhitespace-Whitespace00Count	The file contains an anomalously high number of zero-bytes used as whitespace.
PDFParserWhitespace-Whitespace09Count	The file contains an anomalously high number of 09 bytes used as whitespace.
PDFParserWhitespace-WhitespaceLongestRun	The file contains an anomalously long whitespace area.
PDFParserWhitespace-WhitespaceTotalLength	The file contains an anomalously high number of whitespaces.
PDFParseru3DObjects-NamesAllNames	The file contains an anomalously high number of U3D objects.
PossibleBAT	The file contains evidence of having a standard Windows batch file included. Malware does this to avoid common scanning techniques and to provide persistence.
PossibleDinkumware	The file shows evidence of including some components from DinkumWare. Dinkumware is frequently used in various malware components.
PropertyImpropriety	The file contains suspicious OOXML properties.
RaiseExceptionImports	The file imports functions used to raise exceptions within a program. Malware does this to implement tactics that make standard dynamic code analysis difficult to follow.
ReservedFieldsViolation	The file violates the specification in terms of the use of reserved fields.
ResourceAnomaly	The file contains an anomaly in the resource section. Malware frequently contains malformed or other odd bits in the resource section of a DLL.
RWXSection	This PE may contain modifiable code, which is at best unorthodox and at worst symptomatic of a virus infection. Frequently, this feature implies that the file has been built using something other than a standard compiler or has been modified after it was originally built.
SectorMalfeasance	The file contains structural oddities with OLE sector allocation.
StringInvalid	One of the references to a string in a string table pointed to a negative offset.

Indicator	Description
StringTableNotTerminated	A string table was not terminated with a null byte. This could cause a fault at runtime due to a string that does not end.
StringTruncated	One of the references to a string in a string table pointed to a location after the end of a file.
SuspiciousPDataSection	This PE is hiding something in its "pdata" area, but it is not clear what it is. The "pdata" area in a PE file is generally used for process runtime structures, but this particular file contains something else.
SuspiciousRelocSection	This PE is hiding something in its "relocations" area, but it is not clear what it is. The "relocations" area in a PE file is generally used for relocating particular symbols, but this particular file contains something else.
SuspiciousDirectoryNames	The file contains OLE directory names that are suspicious.
SuspiciousDirectoryStructure	The file has oddities in the OLE directory structure.
SuspiciousEmbedding	The file uses suspicious embedding of OLE.
SuspiciousVBA	The file contains suspicious VBA code.
SuspiciousVBALib	The file shows suspicious VBA library usage.
SuspiciousVBANames	The file contains suspicious names associated with VBA structures.
SuspiciousVBAVersion	The file contains suspicious VBA versioning.
SWFOddity	The file contains certain questionable usages of embedded SWF.
TooMalformedToProcess	The file is so malformed that it could not be parsed completely.
VersionAnomaly	The file has issues with how it presents its version information. Malware does this to avoid detection.

Collection

These indicators represent situations where the file has elements that indicate capabilities or evidence of collecting data. This can include the enumeration of system configuration or the collection of specific sensitive information.

Indicator	Description
BrowserInfoTheft	The file contains evidence of an intent to read passwords stored in browser caches. Malware uses this to collect the passwords for exfiltration.

Indicator	Description
CredentialProvider	The file contains evidence of interaction with a credential provider, or the desire to appear as one. Malware does this because credential providers get access to many types of sensitive data, such as usernames and passwords, and by acting as one, they may be able to subvert the authentication integrity.
CurrentUserInfolImports	The file imports functions that are used to gather information about the currently logged-in user. Malware uses this to determine paths of action to escalate privileges and to better tailor attacks.
DebugStringImports	The file imports functions that are used to output debug strings. Typically, this is disabled in production software, but left on in malware that is being tested.
DiskInfolImports	The file imports functions that can be used to gather details about volumes on the system. Malware uses this in conjunction with listing to determine facts about the volumes in preparation for a further attack.
EnumerateFileImports	The file imports functions that are used to list files. Malware uses this to look for sensitive data or to find further points of attack.
EnumerateModuleImports	The file imports functions that can be used to list all of the DLLs that a running process uses. Malware uses this capability to locate and target specific libraries for loading into a process, and to map out a process it wishes to inject into.
EnumerateNetwork	The file demonstrates evidence of a capability to attempt to enumerate connected networks and network adapters. Malware does this to determine where a target system lies in relation to others, and to look for possible lateral paths.
EnumerateProcessImports	The file imports functions that can be used to list all of the running processes on a system. Malware uses this capability to locate processes to inject into or those that it wishes to delete.
EnumerateVolumelImports	The file imports functions that can be used to list the volumes on the system. Malware uses this to find all the areas that it might need to search for data, or to spread an infection.
GinaImports	The file imports functions that are used to access Gina. Malware does this to attempt to breach the secure ctrl-alt-delete password entry system or other network login functions.
HostnameSearchImports	The file imports functions that are used to gather information about host names on the network and the hostname of the machine itself. Malware uses this capability to better target further attacks or to scan for new targets.
KeystrokeLogImports	The file imports functions that can capture and log keystrokes from the keyboard. Malware uses this to capture and save keystrokes to find sensitive information such as passwords.
OSInfolImports	The file imports functions that are used to gather information about the current operating system. Malware uses this to determine how to better tailor further attacks and to report information back to a controller.

Indicator	Description
PossibleKeylogger	The file contains evidence of key-logger type activity. Malware uses key loggers to collect sensitive information from the keyboard.
PossiblePasswords	The file has evidence of including common passwords, or a structure that would enable brute forcing common passwords. Malware uses this to attempt to penetrate a network further by accessing other resources via password.
ProcessorInfoWMI	The file imports functions that can be used to determine details about the processor. Malware uses this to tailor attacks and to exfiltrate this data to common command-and-control infrastructure.
RDPUsage	The file shows evidence of interacting with the Remote Desktop Protocol (RDP). Malware frequently uses this to move laterally and to offer direct command-and-control functionality.
SpyString	The file is possibly spying on the clipboard or user actions via accessibility API usage.
SystemDirImports	The file imports functions used to locate the system directory. Malware does this to find where many of the installed system binaries are located, as it frequently hides among them.
UserEnvInfoImports	The file imports functions that are used to gather information about the environment of the current logged-in user. Malware uses this to determine details about the logged-in user and to look for other intelligence that can be gleaned from the environment variables.

Data loss

These indicators represent situations where the file has elements that indicate capabilities or evidence of exfiltration of data. This can include outgoing network connections, evidence of acting as a browser, or other network communications.

Indicator	Description
AbnormalNetworkActivity	The file implements a non-standard method of networking. Malware does this to avoid detection of more common networking approaches.
BrowserPluginString	The file has the capability to enumerate or install browser plugins.
ContainsBrowserString	The file contains evidence of attempting to create a custom UserAgent string. Malware frequently uses common UserAgent strings to avoid detection in outgoing requests.
DownloadFileImports	The file imports functions that can be used to download files to the system. Malware uses this as both a way to further stage an attack and to exfiltrate data via the outbound URL.

Indicator	Description
FirewallModifyImports	The file imports functions used to modify the local Windows firewall. Malware uses this to open holes and avoid detection.
HTTPCustomHeaders	The file contains evidence of the creation of other custom HTTP headers. Malware does this to facilitate interactions with command-and-control infrastructures and to avoid detection.
IRCCommands	The file contains evidence of interaction with an IRC server. Malware commonly uses IRC to facilitate a command-and-control infrastructure.
MemoryExfiltrationImports	The file imports functions that can be used to read memory from a running process. Malware uses this to determine proper places to insert itself, or to extract useful information from the memory of a running process, such as passwords, credit cards, or other sensitive information.
NetworkOutboundImports	The file imports functions that can be used to send data out to the network or the general Internet. Malware uses this as a method for exfiltration of data or as a method for command and control.
PipeUsage	The file imports functions that allow the manipulation of named pipes. Malware uses this as a method of communication and of data exfiltration.
RPCUsage	The file imports functions that allow it to interact with Remote Procedure Call (RPC) infrastructure. Malware uses this to spread, or to send data to remote systems for exfiltration.

Deception

These indicators represent situations where the file has elements that indicate capabilities or evidence of a file attempting to be deceptive. Deception can come in the form of hidden sections, inclusion of code to avoid detection, or indications that it is labeled improperly in metadata or other sections.

Indicator	Description
AddedHeader	The file contains an additional, obfuscated PE header that may be a hidden malicious payload.
AddedKernel32	The file contains an additional, obfuscated reference to kernel32.dll, a library that may be used by a malicious payload.
AddedMscoree	The file contains an additional, obfuscated reference to mscoree.dll, a library that may be used by a malicious payload.
AddedMsvbvm	The file contains an additional, obfuscated reference to msvbvm.dll, a library that may be used by a malicious payload compiled for Microsoft Visual Basic 6.

Indicator	Description
AntiVM	The file demonstrates features that can be used to determine if the process is running in a virtual machine. Malware does this to avoid running in virtualized sandboxes that are becoming more common.
AutoitDownloadExecute	The Autolt script can download and execute files. This is often done to deliver additional malicious payloads.
AutoitObfuscationString-Concat	The Autolt script is likely obfuscated with string concatenation. This is often done to avoid detection of whole, suspicious commands.
AutoitShellcodeCalling	The Autolt script uses the CallWindowProc() Windows API function that may indicate the injection of shellcode.
AutoitUseResources	The Autolt script uses data from resources stored alongside the script. Malware often stores important parts of itself as resource data and unpacks them in runtime, and therefore this looks suspicious.
CabinetUsage	The file shows evidence of containing a CAB file. Malware does this to package sensitive components in a way that many detection systems cannot see.
ClearKernel32	The file contains a reference to kernel32.dll, a library that may be used by a malicious payload.
ClearMscoree	The file contains a reference to mscoree.dll, a library that may be used by a malicious payload.
ClearMsvbvm	The file contains a reference to msvbvm.dll, a library that may be used by a malicious payload compiled for Microsoft Visual Basic 6.
ComplexInvalidVersion	The file declares the wrong PDF version.
ComplexJsStenography-Suspected	The file may contain JavaScript code hidden in literal strings.
ContainsEmbeddedDocument	The file contains a document embedded inside the object. Malware can use this to spread an attack to multiple sources or to otherwise hide its true form.
CryptoKeys	The file contains evidence of having an embedded cryptographic key. Malware does this to avoid detection and perhaps as authentication with remote services.
DebugCheckImports	The file imports functions that would allow it to act like a debugger. Malware uses this capability to read and write from other processes.
EmbeddedPE	The PE has additional PEs within it, which is usually only the case with software installation programs. Frequently, malware embeds a PE file that it then drops to disk and executes. This technique is often used to avoid protection scanners by packaging binaries in a format that the underlying scanning technology does not understand.

Indicator	Description
EncodedDosStub1	The PE contains an obfuscated PE DOS stub that may belong to a hidden malicious payload.
EncodedDosStub2	The PE contains an obfuscated PE DOS stub that may belong to a hidden malicious payload.
EncodedPE	The PE has additional PEs hidden within it, which is extremely suspicious. It is similar to the EmbeddedPE indicator, but uses an encoding scheme to attempt to further hide the binary inside the object.
ExecutedDLL	The PE contains evidence of the capability to execute a DLL using common methods. Malware does this as a method to avoid common detection practices.
FakeMicrosoft	The PE claims to be written by Microsoft but it does not look like a Microsoft PE. Malware commonly masquerades as Microsoft PEs to look inconspicuous.
HiddenMachO	The file has another MachO executable file within, which is not properly declared. This may be an attempt to hide the payload from being easily detected.
HTTPCustomUserAgent	The file contains evidence of manipulation of the browser UserAgent. Malware does this to facilitate interactions with command-and-control infrastructures and to avoid detection.
InjectProcessImports	The PE can inject code into other processes. This capability frequently implies that a process is attempting to be deceptive or hostile in some way.
InvisibleEXE	The PE appears to run invisibly, but it is not a background service. It might be designed to remain hidden.
JSTokensSuspicious	The file contains unusually suspicious JavaScript.
MSCertStore	The file shows evidence of interacting with the core Windows certificate store. Malware does this to collect credentials and to insert rogue keys into the stream to facilitate actions such as man-in-the-middle attacks.
MSCryptoImports	The file imports functions to use the core Windows cryptography library. Malware uses this to leverage the locally installed cryptography so that it does not need to carry around its own cryptography.
PDFParserDotDotSlash1-URICount	The file may attempt path traversal using relative paths such as "../".
PDFParserJavaScriptMag-icseval~28	The file may contain obfuscated JavaScript or can run dynamically loaded JavaScript with eval().
PDFParserJavaScriptMag-icsunescape~28	The file may contain obfuscated JavaScript.
PDFParserjsObjectsLength	The file contains an anomalously high number of individual JavaScript scripts.

Indicator	Description
PDFParserJSStreamCount	The file contains an unusually high number of JavaScript-related streams.
PDFParserJSTokenCounts-0cumulativesum	The file contains an anomalously high number of JavaScript tokens.
PDFParserJSTokenCounts-1cumulativesum	The file contains an anomalously high number of JavaScript tokens.
PDFParserNamesAll-NamesSuspicious	The file contains an anomalously high number of suspicious names.
PDFParserNamesObfuscatedNamesSuspicious	The file contains an anomalously high number of obfuscated names.
PDFParserPEdetections	The file contains embedded PE file(s).
PDFParserSwfObjectsxObservationsxSWFObjects-version	The file contains an SWF object with an unusual version number.
PDFParserSwfObjectsxObservationsxSWFObjectsxZLibcmf	The file contains an SWF object with unusual compression parameters.
PDFParserswfObjectsxObservationsxSWFObjectsxZLibflg	The file contains an SWF object with unusual compression flag parameters.
PE_ClearDosStub1	The file contains a DOS stub, indicative of PE file inclusion.
PE_ClearDosStub2	The file contains a DOS stub, indicative of PE file inclusion.
PE_ClearHeader	The file contains PE file header data that does not belong in the file structure.
PEinAppendedSpace	The file contains a PE file that does not belong in the file structure.
PEinFreeSpace	The file contains a PE file that does not belong in the file structure.
ProtectionExamination	The file seems to be looking for common protection systems. Malware does this to initiate an anti-protection action tailored to that installed on the system.
SegmentSuspiciousName	A segment has either an invalid string as a name or an unusual non-standard name. This may indicate post-compilation tampering or the use of packers or obfuscators.
SegmentSuspiciousSize	The segment size is significantly different from the size of all content sections within. This may indicate the use of an unreferenced area or the reservation of space for runtime unpacking of malicious code.

Indicator	Description
SelfExtraction	The file seems to be a self-extracting archive. Malware frequently uses this tactic to obfuscate their true intentions.
ServiceDLL	The file seems to be a service DLL. Service DLLs are loaded in the svchost.exe process and are a common persistence methodology for malware.
StringJsSplitting	The file contains suspicious JS tokens.
SWFinAppendedSpace	The file contains a shockwave flash object that does not belong in the document structure.
TempFileImports	The file imports functions used to access and manipulate temporary files. Malware does this because temporary files tend to avoid detection.
UsesCompression	The file seems to have portions of the code that appear to be compressed. Malware uses these techniques to avoid detection.
VirtualProtectImports	The file imports functions that are used to modify the memory of a running process. Malware does this to inject itself into running processes.
XoredHeader	The file contains an xor-obfuscated PE header that may be a hidden malicious payload.
XoredKernel32	The file contains an xor-obfuscated reference to kernel32.dll, a library that may be used by a malicious payload.
XoredMscoree	The file contains an xor-obfuscated reference to mscoree.dll, a library that may be used by a malicious payload.
XoredMsvbvm	The file contains an xor-obfuscated reference to msvbvm.dll, a library that may be used by a malicious payload compiled for Microsoft Visual Basic 6.

Destruction

These indicators represent situations where the file has elements that indicate capabilities or evidence of destruction. Destructive capabilities include the ability to delete system resources like files or directories.

Indicator	Description
action_writeByte	The VBA script within the file is likely writing bytes to a file, which is an unusual action for a legitimate document.
action_hexToBin	The VBA script within the file is likely using hexadecimal-to-binary conversion that may indicate decoding a hidden malicious payload.
appended_URI	The file contains a link that does not belong in the file structure.

Indicator	Description
appended_exploit	The file contains suspicious data outside of the file structure that may be indicative of an exploit.
appended_macro	The file contains a macro script that does not belong in the file structure.
appended_90_nopsled	The file contains a nop-sled that does not belong in the file structure; this is almost certainly there to facilitate exploitation.
AutorunsPersistence	The file attempts to interact with common methods of persistence (for example, startup scripts). Malware commonly uses these tactics to attain persistence.
DestructionString	The file has capabilities to kill processes or shut down the machine via shell commands.
FileDirDeleteImports	The PE imports functions that can be used to delete files or directories. Malware uses this to break systems and to cover its tracks.
JsHeapSpray	The file likely contains heap spray code.
PossibleLocker	The file demonstrates evidence of a desire to lock out common tools by policy. Malware does this to retain persistence and make detection and cleanup more difficult.
RegistryManipulation	The file imports functions that are used to manipulate the Windows registry. Malware does this to attain persistence, avoid detection, and for many other reasons.
SeBackupPrivilege	The PE might attempt to read files to which it has not been granted access. The SeBackup privilege allows access to files without honoring access controls. It is frequently used by programs that handle backups and is frequently limited to administrative users, but it can be used maliciously to gain access to specific elements that might otherwise be difficult to access.
SeDebugPrivilege	The PE might attempt to tamper with system processes. The SeDebug privilege is used to access processes other than your own and is frequently limited to administrative users. It is often paired with reading and writing to other processes.
SeRestorePrivilege	The PE might attempt to change or delete files to which it has not been granted access. The SeRestore privilege allows writing without consideration of access control.
ServiceControllImports	The file imports functions that can control Windows services on the current system. Malware uses this either to launch itself into the background via installing as a service, or to disable other services that may have a protective function.
SkylinedHeapSpray	The file contains an unmodified version of skylined heap spray code.
SpawnProcessImports	The PE imports functions that can be used to spawn another process. Malware uses this to launch subsequent phases of an infection, typically downloaded from the Internet.

Indicator	Description
StringJsExploit	The file contains JavaScript code that is likely capable of exploitation.
StringJsObfuscation	The file contains JavaScript obfuscation tokens.
TerminateProcessImports	The file imports functions that can be used to stop a running process. Malware uses this to attempt to remove protection systems, or to cause damage to a running system.
trigger_AutoClose	The VBA script within the file is likely trying to execute automatically when the file is closing.
trigger_Auto_Close	The VBA script within the file is likely trying to execute automatically when the file is closing.
trigger_AutoExec	The VBA script within the file is likely trying to execute automatically.
trigger_AutoExit	The VBA script within the file is likely trying to execute automatically when the file is closing.
trigger_AutoNew	The VBA script within the file is likely trying to execute automatically when a new file is being created.
trigger_AutoOpen	The VBA script within the file is likely trying to execute as soon as the file is opened.
trigger_Auto_Open	The VBA script within the file is likely trying to execute as soon as the file is opened.
trigger_DocumentBefore-Close	The VBA script within the file is likely trying to execute automatically just before the file closes.
trigger_DocumentChange	The VBA script within the file is likely trying to execute automatically when the file is being changed.
trigger_Document_Close	The VBA script within the file is likely trying to execute automatically when the file is closing.
trigger_Document_New	The VBA script within the file is likely trying to execute automatically when a new file is being created.
trigger_DocumentOpen	The VBA script within the file is likely trying to execute as soon as the file is opened.
trigger_Document_Open	The VBA script within the file is likely trying to execute as soon as the file is opened.
trigger_NewDocument	The VBA script within the file is likely trying to execute automatically when a new file is being created.

Indicator	Description
trigger_Workbook_Close	The VBA script within the file is likely trying to execute automatically when a Microsoft Excel workbook is closing.
trigger_Workbook_Open	The VBA script within the file is likely trying to execute automatically when a Microsoft Excel workbook is opening.
UserManagementImports	The file imports functions that can be used to change users on the local system. It can add, delete, or change key user details. Malware can use this capability to achieve persistence or cause harm to the local system.
VirtualAllocImports	The file imports functions that are used to create memory in a running process. Malware does this to inject itself into a running process.

Shellcodes

These indicators represent situations where a small piece of code is used as the payload in the exploitation of a software vulnerability. It is called shellcode because it typically starts a command shell from which the attacker can control the compromised machine, but any piece of code that performs a similar task can be called shellcode.

Indicator	Description
ApiHashing	The file contains a byte sequence that looks like shellcode that tries to stealthily find library APIs loaded in memory.
BlackholeV2	The file looks like it might have come from the Blackhole exploit kit.
ComplexGotoEmbed	The file may be able to force the browser to go to an address or to perform an action.
ComplexSuspiciousHeader	The PDF header is located at a non-zero offset which may indicate an attempt to prevent this file from being recognized as a PDF document.
EmbeddedTiff	The file may contain a crafted TIFF image with nop-sled to facilitate exploitation.
EmbeddedXDP	The file likely contains another PDF as an XML Data Package (XDP).
FindKernel32Base1	The file contains a byte sequence that looks like a shellcode that tries to locate kernel32.dll in memory.
FindKernel32Base2	The file contains a byte sequence that looks like a shellcode that tries to locate kernel32.dll in memory.
FindKernel32Base3	The file contains a byte sequence that looks like a shellcode that tries to locate kernel32.dll in memory.
FunctionPrologSig	The file contains a byte sequence that is a typical function prolog, and likely contains shellcode.

Indicator	Description
GetEIP1	The file contains a byte sequence that looks like a shellcode that resolves its own address to locate other things in memory and facilitate exploitation.
GetEIP4	The file contains a byte sequence that looks like a shellcode that resolves its own address to locate other things in memory and facilitate exploitation.
IndirectFnCall1	The file contains a byte sequence that looks like an indirect function call, and is likely shellcode.
IndirectFnCall2	The file contains a byte sequence that looks like an indirect function call, and is likely shellcode.
IndirectFnCall3	The file contains a byte sequence that looks like an indirect function call, and is likely shellcode.
SehSig	The file contains a byte sequence that is typical for Structured Exception Handling (SEH), and likely contains shellcode.
StringLaunchActionBrowser	The file may be able to force the browser to go to an address or to perform an action.
StringLaunchActionShell	The file may be able to execute shell actions.
StringSingExploit	The file might contain an exploit.

Miscellaneous indicators

This section lists the indicators that do not fit into the other categories.

Indicator	Description
AutoitFileOperations	The AutoIt script can perform multiple actions on files. This may be used for information gathering, persistence, or destruction.
AutorunString	The file has the capability to achieve persistence by using autorun mechanisms.
CodepageLookupImports	The file imports functions used to look up the codepage (location) of a running system. Malware uses this to differentiate in which country/region a system is running in to better target particular groups.
MutexImports	The file imports functions to create and manipulate mutex objects. Malware frequently uses mutexes to avoid infecting a system multiple times.
OpenSSLStatic	The file contains a version of OpenSSL compiled to appear stealthy. Malware does this to include cryptography functionality without leaving strong evidence of it.

Indicator	Description
PListString	The file has the capability to interact with property lists that are used by the operating system. This may be used to achieve persistence or to subvert various processes.
PrivEscalationCryptBase	The file shows evidence of attempting to use a privilege escalation using CryptBase. Malware uses this to gain more privileges on the affected system.
ShellCommandString	The file has the capability to use sensitive shell commands for reconnaissance, elevation of privilege, or data destruction.
SystemCallSuspicious	The file has the capability to monitor or control system and other processes, performing debug-like actions.

Appendix: Prometheus monitoring support

Prometheus is a monitoring service for server applications. A Prometheus server uses HTTP GET calls to scrape data from various services, and allows you to run queries against that data. For more information, see <https://prometheus.io/>.

The table below details the metrics that the Cylance Engine provides to a Prometheus server. The Cylance Engine provides the following types of metrics:

- Counter: A metric that can only increase (for example, a total amount).
- Gauge: A counter that can increase or decrease (for example, a count of items in process).
- Histogram: A sample of observations sorted into buckets, along with the sum and count of observations.

Metric	Scope	Type	Description
cyeng_samples_in_process	Global	Gauge	This metric tracks the number of samples that are currently in process. It can range from 0 (idle) up to the max concurrency setting in the INI file or command-line option.
cyeng_total_errors	Global	Counter	This metric is the total number of errors encountered during the scoring process. Composite files (for example, archives) can generate more than one error or a combination of valid, aborted, and error counts.
cyeng_total_unknowns	Global	Counter	This metric is the total number of samples that do not have a corresponding model. Composite files (for example, archives) can result in more unknowns than samples processed, as one archive sample may contain many supported and unsupported files.
cyeng_total_aborted_samples	Global	Counter	This metric is the total number of samples that were aborted due to a timeout or exceeding the maximum nesting level. Composite files (for example, archives) can produce both valid and aborted results.
cyeng_total_bytes_processed	Global	Counter	This metric is the total number of sample bytes that have been processed. This counter is the top-level sample size and does not count samples that are extracted from a sample (for example, Apple Universal Binaries or archives).
cyeng_sample_processing_time	Global	Histogram	This metric is the observation of sample processing times for all sample types. For a composite file, the observation is for the total processing time of all samples it contains.

Metric	Scope	Type	Description
cyeng_sample_size	Global	Histogram	This metric is the observation of the sample size, in bytes. For a composite file, the observation is for the total size of all samples it contains.
cyeng_total_<sample-format>_samples_processed	Per model	Counter	This metric is the total number of processed samples of the type specified with <sample-format>. For a composite file, the count applies to all samples that it contains. For example, if an archive contains another archive, this counter would accumulate 2 counts for the ARC format in addition to any formats inside the archive.
cyeng_total_benign_<sample-format>_samples	Per model	Counter	This metric is the total number of benign samples (a score between 0.0 and +1.0, inclusive) of the type specified with <sample-format>. For a composite file, the count applies to all samples that it contains.
cyeng_total_suspicious_<sample-format>_samples	Per model	Counter	This metric is the total number of suspicious samples (a score between -0.6 and 0.0, exclusive) of the type specified with <sample-format>. For a composite file, the count applies to all samples that it contains.
cyeng_total_malicious_<sample-format>_samples	Per model	Counter	This metric is the total number of malicious samples (a score between -1.0 and -0.6, inclusive) of the type specified with <sample-format>. For a composite file, the count applies to all samples that it contains.
cyeng_<sample-format>_processing_time	Per model	Histogram	This metric is the observation of sample processing times for the sample type specified with <sample-format>. For composite files, the entire processing time of the composite file is included in the composite file's bucket (for example, ARC or MOFAT).

Appendix: CylanceTcpService Protocol

The CylanceTcpService Protocol was known previously as the Infinity Daemon Protocol (IDP).

All requests begin with a single character (byte) specifying the command to perform. The format of the remainder of the request is command-specific and is described below.

Once the service has finished performing a command, it sends a response to the client and then attempts to receive another command from the connection. The format of the response is also command-specific and is described below.

- All numeric fields are represented in binary. For multiple-byte fields such as 16- and 32-bit integers, the byte order is little-endian.
- String fields are encoded as UTF-8 with no initial byte order mark (BOM) or terminating null character. In general, string fields are preceded by a length field that specifies the size of the string in bytes (not characters).

Process command

The process command (`p`) instructs the service to process a file located at the given path. There are two main process commands: Score and Explain; legacy commands ScoreFile, ScoreArchive, ExplainFile, ExplainArchive, and class names are still accepted but not recommended. Class names are explained in [Classless-based and activity-class-based scoring](#).

The format of the `p` command is:

Field	Data element	Description
Command byte	Char	<code>p</code> for the process command.
Command length	Byte	This is the length of the process command, which is normally 5 for the score command or 7 for the explain command. The length is longer when using fully specified class names or passing parameters with the command (for example, passwords for archives).
Command	String	This is the string that contains the command (for example Score or Explain). The field was originally called ActivityClass but that has been deprecated. For more information, see Classless-based and activity-class-based scoring .
File path length	UInt16	This is the length of the file path in bytes. Note: This is not the number of characters in the path but the number of UTF-8 encoded bytes.
File path	String	This is the fully qualified path to the file to process. The file must reside on a path that is visible to the TCP service. Relative paths are relative to the TCP service and should not be used.

The format of the response is:

Field	Data element	Description
Routing tag length	Byte	This is the length of the routing tag. If the command is successful, the length is 0. If an error occurred, then the length is non-zero.
Routing tag	String	This is the routing tag string. If the routing tag length field is 0, this field is empty.
Feature count	Byte	This is the number of features being returned in this response. Up to 255 features can be returned although normally there is only one.
Feature JSON length	UInt32	This is the length of JSON returned for the feature.
Feature JSON	String	This is the JSON object that contains the feature.

The routing tag is a status string returned from the service for every process command. For commands that complete successfully, the routing tag is an empty string (that is, "") which means the first byte of the response is 0. Other routing tags provide additional information.

- "": Success. At least one feature is available in the response. Note that the included feature(s) may indicate an error but the service was able to at least attempt to process the file.
- "unknown": Invalid input or format not supported. This tag is returned when, for example, a file is provided that no loaded model can process. No features were returned as part of this response.
- "error": Unable to process the input. The command should not be repeated and would not succeed if retried (for example, if the file was not found or could not be read). No features were returned as part of this response.
- "fault": A temporary error occurred. The command can be repeated at a later time. Examples of this are file-sharing violations and similar errors. No features were returned as part of this response.

The routing tag only represents the status of the top-level operation and does not report issues during the processing of the file, for instance, once the file has been opened successfully. If an error occurs during the processing of a file, the service still produces a feature that contains more detailed information on the error.

Up to 255 features can be returned in a single process command. For most files, a single feature is returned that contains the score or a set of threat indicators. Due to the existence of archives, multiple features may be returned for a single file (for example, a .zip file contains multiple PE files). When the feature count field is greater than one, the feature JSON length and feature JSON fields are repeated for each feature. Applications must be prepared to examine all features because the order of the features is not guaranteed.

The feature name starts with SampleScoring for a score operation and TTMStatic for an explain operation. Because files may contain other files (for example, archives), a pseudo-path is appended to one of those strings to identify from where the file originated. Each level of nesting is separated by a vertical bar (|).

Due to the maximum length of the feature name (255 bytes), heavily nested feature names may be truncated to fit into the 255-byte limit. Two truncation steps that are applied in order until the length of the feature name fits into the limit:

- Any SHA256 hashes in the name are reduced from 64 characters to 8 characters followed by a tilde (~). SHA256 hashes are used when the name of the file cannot be determined.
- The feature name is the concatenation of "..." with the last 252 bytes of the UTF-8 encoded feature name.

Because the feature name may be truncated, it is recommended to track files by their SHA256 hash that is returned as part of the feature JSON.

For example, a resulting feature-name key for a single file looks like:

```
SampleScoring:example-file.exe
```

Resulting activity keys for an archive look like:

```
SampleScoring:test-file.zip|file1.exe  
SampleScoring:test-file.zip|file2.exe  
SampleScoring:test-file.zip|file3.pdf
```

Archives inside of archives are supported and each level of archive is separated with a vertical bar.

Shutdown command

The shutdown command (`s`) instructs the service to stop listening for incoming connections and exit.

In the `CylanceTcpService` application, the shutdown command is disabled by default. To enable it, add the key-value pair `ShutdownCommand=true` to the `Service` section of the `CylanceTcpService.ini` configuration file or run `CylanceTcpService` with the `-s` or `--shutdown` flag.

The shutdown command is a single byte, "s". There are no other required fields and no response is returned. If the shutdown command is disabled via the `CylanceTcpService.ini` configuration file, the command is simply ignored.

Multiple scores for a file

Because this technology is model based, it is important to apply the correct model to a given file. If the wrong model were used to determine the score, then a file might escape the level of scrutiny that is appropriate for that file. To address this, `CylanceTcpService` checks the file against all model types that have been loaded. As a result, the client must be prepared to receive multiple JSON objects for a given file when using these commands. Each of these objects indicates the model that was used to provide the score found in that object.

Client code should determine what action to take when the service returns more than one score. Depending on the application and workflow, the client application might take into consideration the declared file type (that is, the file extension or the MIME-type). For example, for a given file, if the file extension indicates a PDF file, but the service reports a negative score when analyzing it as a PE file, it is quite possible that it has been purposely disguised as a PDF file to avoid detection.

The `p` command supported by the Infinity Daemon Protocol can return a maximum of 255 results. If a given Score request produces more than 255 results, the list will be truncated. Because a file can be scored with multiple models, this guide recommends no more than 100 files per archive, especially when scoring with two or more models loaded into the service.

Classless-based and activity-class-based scoring

Although `CylanceTcpService` has replaced `TcpShim` and `InfinityTcpService`, it still supports the activity-class-based scoring supported by the legacy applications; however, the newer and more flexible classless-based scoring is highly preferred. In `TcpShim`, the command was the fully-qualified name of the class that should process the file (for example `Cylance.Infinity.Activity.SampleScoring.SampleScoringPEActivity`). This meant that the

client application needed to know the kind of file (in this case, PE) before calling `TcpShim` to process the file. Processing archive files makes this challenge even greater, because a given archive file can contain multiple different file types, each requiring a possibly different activity class.

The `InfinityTcpService` introduced classless-based scoring, in which the class name field in the request was replaced by a command, such as `ScoreFile`, `ScoreArchive`, `ExplainFile`, or `ExplainArchive`. In turn, these commands were further reduced in the `CylanceTcpService` to just `Score` and `Explain`. The client application no longer needs to know beforehand what kind of file it is; the `CylanceTcpService` scores or explains it against all applicable models, and returns zero or more results.

While the `Score` and `Explain` commands are highly recommended, the `ScoreFile`, `ScoreArchive`, `ExplainFile`, and `ExplainArchive` commands still work; however, `ScoreFile` and `ExplainFile` return an error if the file given is an archive, while `ScoreArchive` and `ExplainArchive` return an error if the file is not an archive.

Additionally, the class-based scoring still works but the `ClassName` entry in the activity section(s) of the `CylanceTcpService.ini` configuration file are not populated by default. The class name has also become arbitrary and no longer maps to a name of an activity class in `CylanceTcpService`; it just needs to be a unique string for each activity.

Passwords specified for archives

In the `CylanceTcpService.ini` configuration file, you can specify a set of default passwords that get automatically applied, in order, if the archive cannot be opened. If a specific password is known for an archive, it can be passed as part of the `Command` field. The syntax follows that for URLs:

```
Score?pw=foo
Score?pw=foo&pw=bar
Score?pw=foo,bar
Score?pw=foo,bar&pw=baz
```

One or more passwords can be sent with the command as long as the total length of the command and all passwords does not exceed the 255 bytes allowed for the command. The first password is separated from the command with a `?pw=`. The string following the "=" is a comma-delimited list of passwords. Spaces are not allowed in passwords that are passed via this mechanism.

Alternatively, instead of using a comma-separated list, the `pw` specifier can be repeated multiple times by using `&pw=`. As long as the total command length does not exceed 255 bytes, no limit exists on the number of times that this can be repeated, or how many elements are in one of the comma-separated lists.

The syntax for the `Explain` command is the same except that `Explain` is substituted for `Score`.

File-scoring applications

Client applications, such as `samplescore`, `tmstatic`, and the `InfinityDaemonClient` utility, provide the external application interface to the `CylanceTcpService` file-scoring service. These applications can be called from the shell, or used as a Python API, to invoke directly from client applications.

Samplescore client

The `samplescore` client is an example of a Python-based client for the `samplescored` service script that demonstrates how to analyze files or directories (recursively) and produce a comma-separated-value (CSV) report showing the confidence scores for each file.

Note: The samplescore client works with the Cylance Engine Protocol only. It does not work with the RESTful API.

To use the samplescore client, the samplescored service script must be running.

Samplescore argument	Description	Valid values
-p PORT	This is an optional port number that the samplescored service is listening to.	1024 to 65535
-o PATH FILENAME	This is an optional output path for the CSV report. This is the file name, including the path if necessary, of the file to be scored.	A valid file path A valid file name with read access

The following is an example of the samplescore command:

```
samplescore -p 9002 -o outputfile badfile.exe
```

The samplescore client can score both files and archives with the same command.

The CSV report produced by samplescore provides the following details:

Output	Description
Name	The file name
Threat	How the threat is categorized if the file scores as a bad file
Score	The score returned for the file
Path	The path to the file
Type	The file type
SHA256	The SHA256 hash for the file

Ttmstatic script

The ttmstatic script is an example of a Python script that demonstrates how, using static analysis, to check a file for common indicators of risky software. The script sends the results to the console as text or JSON. To use ttmstatic, the samplescored service script must be running.

Note: The ttmstatic script works with the Cylance Engine Protocol only. It does not work with the RESTful API.

Argument	Description	Valid values
-p PORT	This is an optional port number that the samplescored service is listening to.	1024 to 65535
-j	This argument outputs the results in JSON format rather than the default text.	—

Argument	Description	Valid values
FILENAME	This is the file name, including path if necessary, of the file to be explained.	A valid file name with read access

The following is an example of the `ttmstatic` command:

```
$ ttmstatic -j badfile.exe
```

The `ttmstatic` command can analyze both files and archives using the same command.

InfinityDaemonClient utility

The `InfinityDaemonClient` utility is a simple utility to send commands to the Cylance Engine using the Infinity Daemon Protocol. It does not work when the Cylance Engine is running the Cylance Engine RESTful API (CERA).

Note: The `InfinityDaemonClient` utility works with the Cylance Engine Protocol only. It does not work with the RESTful API.

The basic syntax is:

```
InfinityDaemonClient [<host>:<port>] p <command> <file>
```

The `<host>:<port>` specification is optional and defaults to `localhost:9002`. If the Cylance Engine is running on a different port, then you must specify that port number.

The second argument is the command byte. Only two options are available for this:

Command	Description
p	This command processes a file, either scoring or explaining the file, depending on the <code><command></code> argument. The resulting response from the service is echoed to the terminal. You must include the <code><command></code> and <code><file></code> arguments.
s	If the shutdown command is enabled, this command shuts down the service. When you send a shutdown command, the <code><command></code> and <code><file></code> arguments are not required; if you include them, they are ignored.

The value for the `<command>` argument is either "Score" or "Explain" to score or explain a file or archive, respectively. The legacy commands "ScoreFile", "ScoreArchive", "ExplainFile", and "ExplainArchive", and class-based scoring are still supported but not recommended. For more information, see [Classless-based and activity-class-based scoring](#).

The value for the `<file>` argument is the path to a file or archive. The `InfinityDaemonClient` utility can only process a single file or archive for each invocation.

`InfinityDaemonClient` processes the command and displays the results to the terminal.

The example output of scoring a single file is:

```
$ InfinityDaemonClient localhost:9002 p Score infinityd.exe
Routing tag: ""
1 features document(s)
----SampleScoring:infinityd.exe----
{
  "CentroidHash": "1359238976895146529",
```

```

"SampleFormat": "PE",
"Determinant": "MODEL",
"ModelVersion": "131786662583688997",
"SHA256": "5AE1246EAADE01C5840338850D7B35BF70243FC13A8E006642445DB08CB42A50",
"ParseStatus": "OK",
"Score": 1.0,
"IsComplete": true
}

```

- The first line of the response indicates the status of the operation. An empty routing tag signifies a successful operation. For a full explanation of the routing tag, see [Appendix: CylanceTcpService Protocol](#).
- The second line indicates how many features were generated for the file. For operations on a single file, there is usually only one feature but in some circumstances, multiple features may be produced depending on the type of file.
- The third line indicates that it was a scoring operation (starting with SampleScoring) or an explaining operation (starting with TTMStatic).
- The block within curly braces is a valid JSON object with the results of the operation. While each block is valid JSON, if multiple results were produced, the entire output is not valid JSON. See the archive example below.

Each scoring result includes the following information:

Field	Description
CentroidHash	This field indicates the hash of the centroids currently loaded into the model. A value of 0 indicates that the model had no centroids loaded.
SampleFormat	This field indicates the type of file that was scored.
Determinant	<p>This field specifies where the results were obtained from.</p> <ul style="list-style-type: none"> • MODEL indicates that the score was calculated by the machine-learning model. • BLACKCENTROID and WHITECENTROID indicate that a centroid was hit and the score was changed accordingly. • BLACKLIST and WHITELIST indicate that the file hash was explicitly disallowed (that is, it appeared in the restricted list) or allowed (that is, it appeared in the approved list), respectively. • PARSER specifies an error while parsing the file. • ABORT indicates that the operation was aborted before processing was completed because the scoring took longer than the timeout period to complete. • CONFIG indicates that the maximum nested depth was exceeded when processing an archive, based on a setting from CylanceTcpService.ini configuration file.
ModelVersion	This field indicates the version of the model that produced the score. Because JSON does not handle 64-bit integers well, the version is returned as a string.
SHA256	This field indicates the SHA256 hash of the file.

Field	Description
ParseStatus	<p>This field indicates the status of the parsing of the file.</p> <ul style="list-style-type: none"> • An OK status indicates that the file was parsed successfully. • If the status is not OK, two additional fields, StatusCause and CauseMessage, are included to provide more information about why the file could not be parsed.
Score	<p>This field indicates the score for the file.</p> <ul style="list-style-type: none"> • If an error is generated (the value in the Determinant field is PARSER, CONFIG, or ABORT), this field contains NaN (not a number). • If the value in the Determinant field is WHITECENTROID or WHITELIST, this field is always +1.0. • If the value in the Determinant field is BLACKCENTROID or BLACKLIST, it is always -1.0.
IsComplete	<p>For a single file, this value is always true.</p> <p>When processing an archive, this value indicates whether the archive was completely processed. If the archive has archives inside that exceed the configured maximum nested depth, the value in this field is false to indicate that a partial score was generated, and the value in the Determinant field is CONFIG.</p>

When scoring archives, multiple results are returned. In this example, the test.tar file contains five PE files:

```

$ InfinityDaemonClient localhost:9002 p Score test.tar
Routing tag: ""
6 features document(s)
----SampleScoring:test.tar----
{
  "CentroidHash": "0",
  "SampleFormat": "ARC",
  "Determinant": "MODEL",
  "ModelVersion": "131975059429967678",
  "SHA256": "5D6D21AB0283E17643B64E856D07ACFEBD6FC52EB4B50AFD3CE6891A2A36ECBE",
  "ParseStatus": "OK",
  "Score": 1.0,
  "IsComplete": true
}
----SampleScoring:test.tar|CommonUtils.dll----
{
  "CentroidHash": "1359238976895146529",
  "SampleFormat": "PE",
  "Determinant": "MODEL",
  "ModelVersion": "131786662583688997",
  "SHA256": "7F3FD0F31FA0C6C840D917567670DA3B4A01EF7D64826E7326DEE8B32454296D",
  "ParseStatus": "OK",
  "Score": 1.0,
  "IsComplete": true
}
---- SampleScoring:test.tar|infinityd.exe ----
{
  "CentroidHash": "1359238976895146529",
  "SampleFormat": "PE",
  "Determinant": "MODEL",

```

```

"ModelVersion": "131786662583688997",
"SHA256": "5AE1246EAADE01C5840338850D7B35BF70243FC13A8E006642445DB08CB42A50",
"ParseStatus": "OK",
"Score": 1.0,
"IsComplete": true
}
---- SampleScoring:test.tar|InfinityDotNet.dll ----
{
"CentroidHash": "1359238976895146529",
"SampleFormat": "PE",
"Determinant": "MODEL",
"ModelVersion": "131786662583688997",
"SHA256": "F0A7274835C6D32064ED1D1F09104E881F17ACF544A1ECDF2C430D30D9781EA4",
"ParseStatus": "OK",
"Score": 1.0,
"IsComplete": true
}
---- SampleScoring:test.tar|infinitydt.exe ----
{
"CentroidHash": "1359238976895146529",
"SampleFormat": "PE",
"Determinant": "MODEL",
"ModelVersion": "131786662583688997",
"SHA256": "19F30312D933256BD983DFC6F120F0521D7C97EFB62CB31C5C286F12E4F3C801",
"ParseStatus": "OK",
"Score": 1.0,
"IsComplete": true
}
---- SampleScoring:test.tar|InstallerIDCore.dll ----
{
"CentroidHash": "1359238976895146529",
"SampleFormat": "PE",
"Determinant": "MODEL",
"ModelVersion": "131786662583688997",
"SHA256": "40A3BD9E62336C60DAB2F43E81B8F708882D799D7FAE96746B047B036A3F47F1",
"ParseStatus": "OK",
"Score": 1.0,
"IsComplete": true
}
}

```

For an archive, the first result should be of file format ARC, which indicates that an archive outer container was successfully opened. The score for this result is always +1.0. The IsComplete flag in this top-level result indicates whether the archive was completely explored or not. If the Determinant is PARSER, the archive is somehow corrupt and could not be opened.

The other results follow as described above. The only difference is that the archive name is added to the file path of each result, followed by a vertical bar (|). Multiple levels of archive nesting are each separated by a |. Because the size of the feature-name field is limited, the name may be truncated. For complete details, see [Appendix: CylanceTcpservice Protocol](#).

The Explain command produces threat indicators for a file:

```

$ InfinityDaemonClient localhost:9002 p Score infinityd.exe
Routing tag: ""
1 features document(s)
----TTMStatic:infinityd.exe----
{
  "features": {
    "Collection": {
      "OSInfoImports": true
    }
  }
}

```

```

    },
    "Deception": {
      "ProtectionExamination": true
    }
  },
  "scores": {
    "Destruction": 0,
    "Deception": 5,
    "Collection": 5,
    "DataLoss": 0,
    "Anomalies": 0
  },
  "SampleFormat": "PE"
}

```

The first three lines of the response are the same as for scoring except that SampleScoring is replaced with TTMStatic. Following the header is a JSON block with the threat indicators. It contains three top-level keys.

Field	Description
features	This field indicates the collection of threat-indicator features discovered in the file. The contents of this block change based on the features in the file. The keys are the category of the feature with the value of the threat indicator indexed by the threat-indicator name.
scores	This field indicates the total count of threat indicators for each category.
SampleFormat	This field indicates the format of the file.

For a complete list of the supported threat indicators and their categories, see [Appendix: Threat indicators](#).

When explaining an archive, the TTMStatic header begins each result with a path with the name of the archive (or archives, if nested) separated by a vertical bar (|). Because the size of the feature-name field is limited, the name may be truncated. For complete details on how feature names are truncated, see [Appendix: CylanceTcpService Protocol](#).

Legal notice

©2024 BlackBerry Limited. Trademarks, including but not limited to BLACKBERRY, BBM, BES, EMBLEM Design, ATHOC, CYLANCE and SECUSMART are the trademarks or registered trademarks of BlackBerry Limited, its subsidiaries and/or affiliates, used under license, and the exclusive rights to such trademarks are expressly reserved. All other trademarks are the property of their respective owners.

Patents, as applicable, identified at: www.blackberry.com/patents.

This documentation including all documentation incorporated by reference herein such as documentation provided or made available on the BlackBerry website provided or made accessible "AS IS" and "AS AVAILABLE" and without condition, endorsement, guarantee, representation, or warranty of any kind by BlackBerry Limited and its affiliated companies ("BlackBerry") and BlackBerry assumes no responsibility for any typographical, technical, or other inaccuracies, errors, or omissions in this documentation. In order to protect BlackBerry proprietary and confidential information and/or trade secrets, this documentation may describe some aspects of BlackBerry technology in generalized terms. BlackBerry reserves the right to periodically change information that is contained in this documentation; however, BlackBerry makes no commitment to provide any such changes, updates, enhancements, or other additions to this documentation to you in a timely manner or at all.

This documentation might contain references to third-party sources of information, hardware or software, products or services including components and content such as content protected by copyright and/or third-party websites (collectively the "Third Party Products and Services"). BlackBerry does not control, and is not responsible for, any Third Party Products and Services including, without limitation the content, accuracy, copyright compliance, compatibility, performance, trustworthiness, legality, decency, links, or any other aspect of Third Party Products and Services. The inclusion of a reference to Third Party Products and Services in this documentation does not imply endorsement by BlackBerry of the Third Party Products and Services or the third party in any way.

EXCEPT TO THE EXTENT SPECIFICALLY PROHIBITED BY APPLICABLE LAW IN YOUR JURISDICTION, ALL CONDITIONS, ENDORSEMENTS, GUARANTEES, REPRESENTATIONS, OR WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY CONDITIONS, ENDORSEMENTS, GUARANTEES, REPRESENTATIONS OR WARRANTIES OF DURABILITY, FITNESS FOR A PARTICULAR PURPOSE OR USE, MERCHANTABILITY, MERCHANTABLE QUALITY, NON-INFRINGEMENT, SATISFACTORY QUALITY, OR TITLE, OR ARISING FROM A STATUTE OR CUSTOM OR A COURSE OF DEALING OR USAGE OF TRADE, OR RELATED TO THE DOCUMENTATION OR ITS USE, OR PERFORMANCE OR NON-PERFORMANCE OF ANY SOFTWARE, HARDWARE, SERVICE, OR ANY THIRD PARTY PRODUCTS AND SERVICES REFERENCED HEREIN, ARE HEREBY EXCLUDED. YOU MAY ALSO HAVE OTHER RIGHTS THAT VARY BY STATE OR PROVINCE. SOME JURISDICTIONS MAY NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES AND CONDITIONS. TO THE EXTENT PERMITTED BY LAW, ANY IMPLIED WARRANTIES OR CONDITIONS RELATING TO THE DOCUMENTATION TO THE EXTENT THEY CANNOT BE EXCLUDED AS SET OUT ABOVE, BUT CAN BE LIMITED, ARE HEREBY LIMITED TO NINETY (90) DAYS FROM THE DATE YOU FIRST ACQUIRED THE DOCUMENTATION OR THE ITEM THAT IS THE SUBJECT OF THE CLAIM.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN YOUR JURISDICTION, IN NO EVENT SHALL BLACKBERRY BE LIABLE FOR ANY TYPE OF DAMAGES RELATED TO THIS DOCUMENTATION OR ITS USE, OR PERFORMANCE OR NON-PERFORMANCE OF ANY SOFTWARE, HARDWARE, SERVICE, OR ANY THIRD PARTY PRODUCTS AND SERVICES REFERENCED HEREIN INCLUDING WITHOUT LIMITATION ANY OF THE FOLLOWING DAMAGES: DIRECT, CONSEQUENTIAL, EXEMPLARY, INCIDENTAL, INDIRECT, SPECIAL, PUNITIVE, OR AGGRAVATED DAMAGES, DAMAGES FOR LOSS OF PROFITS OR REVENUES, FAILURE TO REALIZE ANY EXPECTED SAVINGS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, LOSS OF BUSINESS OPPORTUNITY, OR CORRUPTION OR LOSS OF DATA, FAILURES TO TRANSMIT OR RECEIVE ANY DATA, PROBLEMS ASSOCIATED WITH ANY APPLICATIONS USED IN CONJUNCTION WITH BLACKBERRY PRODUCTS OR SERVICES, DOWNTIME COSTS, LOSS OF THE USE OF BLACKBERRY PRODUCTS OR SERVICES OR ANY PORTION THEREOF OR OF ANY AIRTIME SERVICES, COST OF SUBSTITUTE GOODS, COSTS OF COVER, FACILITIES OR SERVICES, COST OF CAPITAL, OR OTHER SIMILAR PECUNIARY LOSSES, WHETHER OR NOT SUCH DAMAGES

WERE FORESEEN OR UNFORESEEN, AND EVEN IF BLACKBERRY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN YOUR JURISDICTION, BLACKBERRY SHALL HAVE NO OTHER OBLIGATION, DUTY, OR LIABILITY WHATSOEVER IN CONTRACT, TORT, OR OTHERWISE TO YOU INCLUDING ANY LIABILITY FOR NEGLIGENCE OR STRICT LIABILITY.

THE LIMITATIONS, EXCLUSIONS, AND DISCLAIMERS HEREIN SHALL APPLY: (A) IRRESPECTIVE OF THE NATURE OF THE CAUSE OF ACTION, DEMAND, OR ACTION BY YOU INCLUDING BUT NOT LIMITED TO BREACH OF CONTRACT, NEGLIGENCE, TORT, STRICT LIABILITY OR ANY OTHER LEGAL THEORY AND SHALL SURVIVE A FUNDAMENTAL BREACH OR BREACHES OR THE FAILURE OF THE ESSENTIAL PURPOSE OF THIS AGREEMENT OR OF ANY REMEDY CONTAINED HEREIN; AND (B) TO BLACKBERRY AND ITS AFFILIATED COMPANIES, THEIR SUCCESSORS, ASSIGNS, AGENTS, SUPPLIERS (INCLUDING AIRTIME SERVICE PROVIDERS), AUTHORIZED BLACKBERRY DISTRIBUTORS (ALSO INCLUDING AIRTIME SERVICE PROVIDERS) AND THEIR RESPECTIVE DIRECTORS, EMPLOYEES, AND INDEPENDENT CONTRACTORS.

IN ADDITION TO THE LIMITATIONS AND EXCLUSIONS SET OUT ABOVE, IN NO EVENT SHALL ANY DIRECTOR, EMPLOYEE, AGENT, DISTRIBUTOR, SUPPLIER, INDEPENDENT CONTRACTOR OF BLACKBERRY OR ANY AFFILIATES OF BLACKBERRY HAVE ANY LIABILITY ARISING FROM OR RELATED TO THE DOCUMENTATION.

Prior to subscribing for, installing, or using any Third Party Products and Services, it is your responsibility to ensure that your airtime service provider has agreed to support all of their features. Some airtime service providers might not offer Internet browsing functionality with a subscription to the BlackBerry® Internet Service. Check with your service provider for availability, roaming arrangements, service plans and features. Installation or use of Third Party Products and Services with BlackBerry's products and services may require one or more patent, trademark, copyright, or other licenses in order to avoid infringement or violation of third party rights. You are solely responsible for determining whether to use Third Party Products and Services and if any third party licenses are required to do so. If required you are responsible for acquiring them. You should not install or use Third Party Products and Services until all necessary licenses have been acquired. Any Third Party Products and Services that are provided with BlackBerry's products and services are provided as a convenience to you and are provided "AS IS" with no express or implied conditions, endorsements, guarantees, representations, or warranties of any kind by BlackBerry and BlackBerry assumes no liability whatsoever, in relation thereto. Your use of Third Party Products and Services shall be governed by and subject to you agreeing to the terms of separate licenses and other agreements applicable thereto with third parties, except to the extent expressly covered by a license or other agreement with BlackBerry.

The terms of use of any BlackBerry product or service are set out in a separate license or other agreement with BlackBerry applicable thereto. NOTHING IN THIS DOCUMENTATION IS INTENDED TO SUPERSEDE ANY EXPRESS WRITTEN AGREEMENTS OR WARRANTIES PROVIDED BY BLACKBERRY FOR PORTIONS OF ANY BLACKBERRY PRODUCT OR SERVICE OTHER THAN THIS DOCUMENTATION.

BlackBerry Enterprise Software incorporates certain third-party software. The license and copyright information associated with this software is available at <http://worldwide.blackberry.com/legal/thirdpartysoftware.jsp>.

BlackBerry Limited
2200 University Avenue East
Waterloo, Ontario
Canada N2K 0A7

BlackBerry UK Limited
Ground Floor, The Pearce Building, West Street,
Maidenhead, Berkshire SL6 1RL
United Kingdom

Published in Canada