
Lastline Analyst API Documentation

Release 2.0

Lastline, Inc.

Oct 27, 2021

CONTENTS

| | | |
|----------|--|------------|
| 1 | Overview | 1 |
| 1.1 | Supported Artifacts | 1 |
| 1.2 | Getting Started | 12 |
| 1.3 | API Concepts | 13 |
| 1.4 | Workflow | 13 |
| 1.5 | Handling of Containers | 13 |
| 2 | API Reference | 15 |
| 2.1 | Authentication | 15 |
| 2.2 | Response Format | 15 |
| 2.3 | Methods | 16 |
| 2.4 | Error Codes | 45 |
| 2.5 | Submission Metadata | 47 |
| 2.6 | Web-Portal Integration | 47 |
| 3 | Analysis Results | 49 |
| 3.1 | Analysis Report Format | 51 |
| 3.2 | Report Format | 52 |
| 3.3 | Report Format <i>ll-int-win</i> | 57 |
| 3.4 | Report Format <i>ll-int-osx</i> | 75 |
| 3.5 | Report Format <i>ll-win-timeline-based</i> | 79 |
| 3.6 | PE Stats information | 81 |
| 3.7 | PE Resource Stats information | 81 |
| 3.8 | Report Format <i>ll-osx-timeline-based</i> | 91 |
| 3.9 | Report Format <i>ll-int-win-doc</i> | 96 |
| 3.10 | Report Format <i>ll-int-apk</i> | 96 |
| 3.11 | Report Format <i>ll-int-archive</i> | 101 |
| 3.12 | Report Format <i>ll-web</i> | 102 |
| 3.13 | Report Format <i>ll-static</i> | 111 |
| 3.14 | Report Format <i>ll-ioc-json</i> | 118 |
| 3.15 | Report Format <i>ll-pcap</i> | 120 |
| 3.16 | Report Format <i>ll-flash</i> | 122 |
| 3.17 | Report Format <i>ll-doc</i> | 125 |
| 3.18 | Report Descriptions | 127 |
| 4 | Child Tasks | 129 |
| 5 | Sample API Clients | 131 |
| 5.1 | Analyst API client | 131 |
| 5.2 | Analysis Client Shell | 146 |

| | | |
|-----|-------------------------------------|------------|
| 5.3 | Analyst API Shell Example | 147 |
| 5.4 | Analyst API Shell Helpers | 151 |
| 5.5 | Application Bundle Module | 153 |
| | Python Module Index | 159 |
| | Index | 161 |

OVERVIEW

The Lastline Analyst API provides functionality for submitting resources for analysis and obtaining the results. Currently, it supports URLs as well as various types of executables and documents.

Executables are analyzed by running them inside a sandbox, recording the behavior of the program, and classifying the file based on the observed actions. Similarly, documents are opened in an instrumented file-editor/viewer or by analyzing any active components (such as scripts) embedded inside the documents; in either case, the behavior of the code is used for detecting if the file contains any anomalies.

Additionally, the content of a submitted file is analyzed for structural similarities with other, previously classified malware artifacts.

URLs are analyzed by visiting them with special, instrumented browsers and observing actions inside the browser or its interactions with its environment.

The latest version of this documentation can be found at <https://analysis.lastline.com/analysis/api-docs/html/overview.html>, or downloaded in PDF format from <https://analysis.lastline.com/analysis/api-docs/LastlineAnalystAPI.pdf>.

1.1 Supported Artifacts

The API supports submissions of URLs and files. The maximum file size is 64 MB for the hosted Lastline infrastructure - for *On-Premises* deployments, the limit is configurable (up to 100MB) and defaults to 10 MB.

The following table provides an overview of the supported file types:

- **AceArchiveFile:** ACE archive data
Lastline mime type: *application/x-ace*
Typical extension: *.ace*
- **BzipArchiveFile:** bzip2 compressed data
Lastline mime type: *application/x-bzip*
Typical extensions: *.bz, .bz2, .tbz, .tbz2*
- **CabArchiveFile:** Microsoft Cabinet archive data
Lastline mime type: *application/vnd.ms-cab-compressed*
Typical extension: *.cab*
- **DiagCabArchiveFile:** Microsoft Diagnostic Cabinet archive data
Lastline mime type: *application/vnd.ms-diagcab-compressed*
Typical extension: *.diagcab*

- **DmgArchiveFile:** Apple disk image
Lastline mime type: *application/x-apple-diskimage*
Typical extensions: *.dmg, .smi*
- **Rfc2822EmailArchiveFile:** RFC2822-formatted Email file
Lastline mime type: *data/email-rfc2822*
Typical extension: *.eml*
- **GzipArchiveFile:** gzip compressed data
Lastline mime type: *application/x-gzip*
Typical extensions: *.gz, .tgz*
- **JarArchiveFile:** Java JAR archive
Lastline mime type: *application/java-archive*
Typical extension: *.jar*
- **WebappJarArchiveFile:** Java Webapp archive
Lastline mime type: *application/war-archive*
Typical extension: *.war*
- **LhaArchiveFile:** LHa archive data
Lastline mime type: *application/x-lha*
Typical extensions: *.lha, .lzh*
- **LzmaArchiveFile:** LZMA compressed data
Lastline mime type: *application/x-lzma*
Typical extension: *.lzma*
- **NugetArchiveFile:** NuGet package archive
Lastline mime type: *application/x-nuget*
Typical extension: *.nupkg*
- **UDFISOArchiveFile:** UDF filesystem data
Lastline mime type: *application/x-udf-image*
Typical extensions: *.iso, .udf*
- **ISO9660ISOArchiveFile:** ISO 9660 CD-ROM filesystem data
Lastline mime type: *application/x-iso9660-image*
Typical extension: *.iso*
- **RarArchiveFile:** RAR archive data
Lastline mime type: *application/x-rar*
Typical extension: *.rar*
- **Rar5ArchiveFile:** RAR archive data, version 5
Lastline mime type: *application/x-rar5*
Typical extension: *.rar*

- **TarArchiveFile**: POSIX tar archive data
Lastline mime type: *application/tar*
Typical extension: *.tar*
- **DocumentLLAppBundleTarArchiveFile**: Lastline Application Bundle Document Type
Lastline mime type: *application/llappbundle-document*
Typical extensions: *.tar, .llappbundle, .llapp*
- **WindowsExecutableLLAppBundleTarArchiveFile**: Lastline Application Bundle Windows Executable Type
Lastline mime type: *application/llappbundle-windows-executable*
Typical extensions: *.tar, .llappbundle, .llapp*
- **WebReplayLLAppBundleTarArchiveFile**: Lastline Application Bundle Web Replay Type
Lastline mime type: *application/llappbundle-web-replay*
Typical extensions: *.tar, .llappbundle, .llapp*
- **TnefArchiveFile**: Transport Neutral Encapsulation Format
Lastline mime type: *application/vnd.ms-tnef*
Typical extension: *.dat*
- **XarArchiveFile**: XAR archive data
Lastline mime type: *application/x-xar*
Typical extensions: *.xar, .pkg*
- **XzArchiveFile**: XZ compressed data
Lastline mime type: *application/x-xz*
Typical extensions: *.xz, .txz*
- **ZipArchiveFile**: Zip archive data
Lastline mime type: *application/zip*
Typical extension: *.zip*
- **SevenZipArchiveFile**: 7-zip archive data
Lastline mime type: *application/x-7z-compressed*
Typical extension: *.7z*
- **MicrosoftSettingContentDataFile**: Microsoft Content-Settings data file *
Lastline mime type: *text/ms-settingcontent*
Typical extension: *.settingcontent-ms*
- **CsvDataFile**: CSV Data
Lastline mime type: *data/csv*
Typical extension: *.csv*
- **InternetInquiryDataFile**: Internet Inquiry data file *
Lastline mime type: *text/x-ms-iqy*
Typical extension: *.iqy*

- **SymbolicLinkDataFile:** Symbolic Link data file
Lastline mime type: *data/symbolic-link*
Typical extensions: *.slk, .sylvk*
- **PcapDataFile:** tcpdump capture file
Lastline mime type: *application/vnd.tcpdump.pcap*
Typical extensions: *.pcap, .pcapng*
- **WordHangulCdfDocFile:** Hangul Word Processor document
Lastline mime type: *application/hangul-word*
Typical extension: *.hwp*
- **ChmDocFile:** Microsoft Windows HtmlHelp data
Lastline mime type: *application/x-chm*
Typical extension: *.chm*
- **HangulDocFile:** Hangul HWP3/HWP2000 document *
Lastline mime type: *application/x-hwp*
Typical extension: *.hwp*
- **ExcelMsMimeDocFile:** Microsoft Excel document in MHTML format
Lastline mime type: *application/msoffice-mime-xls*
Typical extension: *.xls*
- **PowerpointMsMimeDocFile:** Microsoft Powerpoint document in MHTML format
Lastline mime type: *application/msoffice-mime-ppt*
Typical extension: *.ppt*
- **WordMsMimeDocFile:** Microsoft Word document in MHTML format
Lastline mime type: *application/msoffice-mime-doc*
Typical extension: *.doc*
- **ExcelMsDocFile:** Microsoft Office Excel document
Lastline mime type: *application/msoffice-xls*
Typical extension: *.xls*
- **TemplateExcelMsDocFile:** Microsoft Office Excel template document
Lastline mime type: *application/msoffice-xlt*
Typical extension: *.xlt*
- **ExcelEncryptedKnownMsDocFile:** Microsoft Office Excel document (with password)
Lastline mime type: *application/msoffice-xls-encrypted*
Typical extensions: *.xls, .xlsx*
- **MacroExcelEncryptedKnownMsDocFile:** Microsoft Office Excel document (with password), with macros
Lastline mime type: *application/msoffice-xlam-encrypted*
Typical extension: *.xlam*

- **PowerpointEncryptedKnownMsDocFile:** Microsoft Office Powerpoint document (with password)
Lastline mime type: *application/msoffice-ppt-encrypted*
Typical extensions: *.ppt, .pptx*
- **WordEncryptedKnownMsDocFile:** Microsoft Office Word document (with password)
Lastline mime type: *application/msoffice-doc-encrypted*
Typical extensions: *.doc, .docx*
- **PowerpointMsDocFile:** Microsoft Office Powerpoint document
Lastline mime type: *application/msoffice-ppt*
Typical extensions: *.ppt, .pps*
- **TemplatePowerpointMsDocFile:** Microsoft Office Powerpoint template document
Lastline mime type: *application/msoffice-pot*
Typical extension: *.pot*
- **WordMsDocFile:** Microsoft Office Word document
Lastline mime type: *application/msoffice-doc*
Typical extension: *.doc*
- **PublisherWordMsDocFile:** Microsoft Publisher document
Lastline mime type: *application/msoffice-publisher*
Typical extension: *.pub*
- **TemplateWordMsDocFile:** Microsoft Office Word document template
Lastline mime type: *application/msoffice-dot*
Typical extension: *.dot*
- **OoDocFile:** Open/LibreOffice document
Lastline mime type: *application/vnd.oasis.opendocument*
Typical extensions: *.odp, .otp, .ods, .odt, .ott, .odg, .otg*
- **PdfDocFile:** PDF document
Lastline mime type: *application/pdf*
Typical extension: *.pdf*
- **WordPerfectDocFile:** WordPerfect document
Lastline mime type: *application/wordperfect*
Typical extension: *.wpd*
- **RtfDocFile:** RTF document
Lastline mime type: *text/rtf*
Typical extension: *.rtf*
- **SwfDocFile:** Macromedia Flash data
Lastline mime type: *application/x-shockwave-flash*
Typical extension: *.swf*

- **ExcelXmlDocFile:** XML-based Microsoft Office Excel document, pre-Office2007
Lastline mime type: *application/x-spreadsheetml*
Typical extension: *.xml*
- **PowerpointXmlDocFile:** XML-based Microsoft Office Powerpoint presentation, pre-Office2007
Lastline mime type: *application/x-presentationml*
Typical extension: *.xml*
- **WordXmlDocFile:** XML-based Microsoft Office Word document, pre-Office2007
Lastline mime type: *application/x-wordprocessingml*
Typical extension: *.xml*
- **XdpXmlDocFile:** Adobe XDP document
Lastline mime type: *application/vnd.adobe.xdp+xml*
Typical extension: *.xdp*
- **XslXmlDocFile:** eXtensible Stylesheet Language for XML file
Lastline mime type: *text/xsl*
Typical extension: *.xsl*
- **ExcelMsDocxFile:** Microsoft Office Excel document, Office Open XML format
Lastline mime type: *application/msoffice-xlsx*
Typical extension: *.xlsx*
- **MacroExcelMsDocxFile:** Microsoft Office Excel document, Office Open XML format, with macros
Lastline mime type: *application/msoffice-xlsm*
Typical extension: *.xlsm*
- **BinaryMacroExcelMsDocxFile:** Microsoft Office Excel document, Office Open XML format, with macros and binary storage
Lastline mime type: *application/msoffice-xlsb*
Typical extension: *.xlsb*
- **TemplateExcelMsDocxFile:** Microsoft Office Excel template document, Office Open XML format
Lastline mime type: *application/msoffice-rltx*
Typical extension: *.rltx*
- **MacroTemplateExcelMsDocxFile:** Microsoft Office Excel spreadsheet template, Office Open XML format, with macros
Lastline mime type: *application/msoffice-rltm*
Typical extension: *.rltm*
- **PowerpointMsDocxFile:** Microsoft Office Powerpoint document, Office Open XML format
Lastline mime type: *application/msoffice-pptx*
Typical extensions: *.pptx, .ppsx*
- **MacroAddInPowerpointMsDocxFile:** Microsoft Office Powerpoint AddIn document, Office Open XML format, with macros

Lastline mime type: *application/msoffice-ppam*

Typical extension: *.ppam*

- **MacroPowerpointMsDocxFile:** Microsoft Office Powerpoint document, Office Open XML format, with macros

Lastline mime type: *application/msoffice-pptm*

Typical extension: *.pptm*

- **SlideshowPowerpointMsDocxFile:** Microsoft Office Powerpoint Slideshow, Office Open XML format

Lastline mime type: *application/msoffice-ppsx*

Typical extension: *.ppsx*

- **MacroSlideshowPowerpointMsDocxFile:** Microsoft Office Powerpoint Slideshow, Office Open XML format, with macros

Lastline mime type: *application/msoffice-ppsm*

Typical extension: *.ppsm*

- **TemplatePowerpointMsDocxFile:** Microsoft Office Powerpoint template document, Office Open XML format

Lastline mime type: *application/msoffice-potx*

Typical extension: *.potx*

- **MacroTemplatePowerpointMsDocxFile:** Microsoft Office Powerpoint presentation template, Office Open XML format, with macros

Lastline mime type: *application/msoffice-potm*

Typical extension: *.potm*

- **WordMsDocxFile:** Microsoft Office Word document, Office Open XML format

Lastline mime type: *application/msoffice-docx*

Typical extension: *.docx*

- **MacroWordMsDocxFile:** Microsoft Office Word document, Office Open XML format, with macros

Lastline mime type: *application/msoffice-docm*

Typical extension: *.docm*

- **TemplateWordMsDocxFile:** Microsoft Office Word template document, Office Open XML format

Lastline mime type: *application/msoffice-dotx*

Typical extension: *.dotx*

- **MacroTemplateWordMsDocxFile:** Microsoft Office Word document template, Office Open XML format, with macros

Lastline mime type: *application/msoffice-dotm*

Typical extension: *.dotm*

- **MsXpsMsDocxFile:** Microsoft XPS document

Lastline mime type: *application/vnd.ms-xpsdocument*

Typical extension: *.xps*

- **OpenXpsMsDocxFile:** OpenXPS document

Lastline mime type: *application/oxps*

Typical extension: *.oxps*

- **JavaClassExeFile**: compiled Java class data *

Lastline mime type: *application/x-java-class*

Typical extension: *.class*

- **ComExeFile**: COM executable for DOS

Lastline mime type: *application/x-com*

Typical extension: *.com*

- **EicarComExeFile**: EICAR test virus

Lastline mime type: *application/x-eicar*

Typical extension: *.com*

- **DosExeFile**: MS-DOS executable *

Lastline mime type: *application/x-dosexec*

Typical extension: *.exe*

- **ElfExeFile**: ELF executable

Lastline mime type: *application/x-elf*

Typical extension: *.elf*

- **MsInstallerExeFile**: Microsoft Installer file *

Lastline mime type: *application/x-msi*

Typical extension: *.msi*

- **LnkExeFile**: Microsoft Windows shortcut

Lastline mime type: *application/x-ms-shortcut*

Typical extensions: *.lnk, .url*

- **MachOExeFile**: Mach-O executable

Lastline mime type: *application/x-mach-o-binary*

Typical extensions: *.o, .dylib, .bundle*

- **BundleMachOExeFile**: Mach-O executable bundle

Lastline mime type: *application/x-mach-o-binary-bundle*

Typical extension: *.bundle*

- **ExecutableMachOExeFile**: Mach-O executable program

Lastline mime type: *application/x-mach-o-binary-executable*

Typical extension: *.o*

- **LibraryMachOExeFile**: Mach-O executable library

Lastline mime type: *application/x-mach-o-binary-library*

Typical extensions: *.o, .dylib*

- **PeExeFile**: PE executable

Lastline mime type: *application/x-pe*

Typical extensions: *.exe, .scr, .pif, .com, .bat, .cmd, .cpl*

- **RarSfxPeExeFile**: RAR SFX PE executable

Lastline mime type: *application/x-rar-sfx-pe*

Typical extension: *.exe*

- **ZipSfxPeExeFile**: Zip SFX PE executable

Lastline mime type: *application/x-zip-sfx-pe*

Typical extension: *.exe*

- **SevenZipSfxPeExeFile**: 7zip SFX PE executable

Lastline mime type: *application/x-7zip-sfx-pe*

Typical extension: *.exe*

- **LastlineTestPeExeFile**: Lastline PE test file

Lastline mime type: *application/x-lastline-test*

Typical extensions: *.exe, .dll, .sys*

- **MachOFatUniversalExeFile**: Mach-O fat file

Lastline mime type: *application/x-mach-o-fat-binary*

Typical extensions: *.o, .dylib, .bundle*

- **TiffImageFile**: TIFF image data

Lastline mime type: *image/tiff*

Typical extensions: *.tif, .tiff*

- **SvgXmlImageFile**: SVG image data *

Lastline mime type: *image/svg*

Typical extension: *.svg*

- **HTAScriptFile**: HTA Script File text *

Lastline mime type: *text/hta*

Typical extension: *.hta*

- **VBAVisualBasicScriptFile**: Visual Basic for Applications text *

Lastline mime type: *text/vba*

Typical extension: *.vba*

- **VBSVisualBasicScriptFile**: VBScript text *

Lastline mime type: *text/vbscript*

Typical extension: *.vbs*

- **EncodedVBSVisualBasicScriptFile**: VBScript encoded script *

Lastline mime type: *application/encodedvbscript*

Typical extension: *.vbe*

- **BatchScriptFile**: Batch script text *

Lastline mime type: *text/x-msdos-batch*

Typical extensions: *.bat, .cmd*

- **JavascriptScriptFile**: JavaScript text

Lastline mime type: *application/javascript*

Typical extension: *.js*

- **EncodedJavascriptScriptFile**: JScript encoded script *

Lastline mime type: *application/encodedjscript*

Typical extension: *.jse*

- **PerlScriptFile**: Perl script text

Lastline mime type: *text/x-perl*

Typical extensions: *.pl, .pm*

- **PowershellScriptFile**: PowerShell text *

Lastline mime type: *text/x-powershell*

Typical extensions: *.ps1, .psm1, .psd1*

- **PythonScriptFile**: Python script text

Lastline mime type: *text/x-python*

Typical extension: *.py*

- **ShellScriptFile**: Shell script text

Lastline mime type: *text/x-shellscript*

Typical extensions: *.sh, .command*

- **WindowsScriptFile**: Windows Script File text

Lastline mime type: *text/x-wsf*

Typical extension: *.wsf*

- **InternetShortcutFile**: Internet Shortcut file

Lastline mime type: *text/x-internetshortcut*

Typical extensions: *.url, .website*

- **HtmlTextFile**: HTML document

Lastline mime type: *text/html*

Typical extensions: *.html, .htm*

* These file types are **only** supported if the Windows sandbox is configured for the requesting license. **NOTE:** In some cases, the *Lastline mime types* shown in the above list represent a unified, generalized version of standard mime types. This allows mapping different, semantically equivalent types into a single type.

The API supports the most common container formats. When submitting an container (archive or ISO, for example) file, the API will automatically attempt to extract and analyze the contained files. More precisely, the API will create a *child analysis* (for details, see [Child Tasks](#)) for files extracted from the container that have a supported file type. Additionally, for multi-file containers containing executables (such as programs or scripts) that should be analyzed as whole, the API attempts to generate *program bundles* (see [Handling of Containers](#)). For encrypted containers (such as encrypted archives), [submit_file\(\)](#) allows you to specify a decryption password or list of potential passwords - if none is specified, the API attempts decryption using common industry-standard passwords (such as “infected”).

The API supports the following container types:

- **AceArchiveFile**: ACE archive data
 - Lastline mime type: *application/x-ace*
 - Typical extension: *.ace*
- **BzipArchiveFile**: bzip2 compressed data
 - Lastline mime type: *application/x-bzip*
 - Typical extensions: *.bz, .bz2, .tbz, .tbz2*
- **CabArchiveFile**: Microsoft Cabinet archive data
 - Lastline mime type: *application/vnd.ms-cab-compressed*
 - Typical extension: *.cab*
- **DiagCabArchiveFile**: Microsoft Diagnostic Cabinet archive data
 - Lastline mime type: *application/vnd.ms-diagcab-compressed*
 - Typical extension: *.diagcab*
- **GzipArchiveFile**: gzip compressed data
 - Lastline mime type: *application/x-gzip*
 - Typical extensions: *.gz, .tgz*
- **LhaArchiveFile**: LHa archive data
 - Lastline mime type: *application/x-lha*
 - Typical extensions: *.lha, .lzh*
- **LzmaArchiveFile**: LZMA compressed data
 - Lastline mime type: *application/x-lzma*
 - Typical extension: *.lzma*
- **NugetArchiveFile**: NuGet package archive
 - Lastline mime type: *application/x-nuget*
 - Typical extension: *.nupkg*
- **UDFISOArchiveFile**: UDF filesystem data
 - Lastline mime type: *application/x-udf-image*
 - Typical extensions: *.iso, .udf*
- **ISO9660ISOArchiveFile**: ISO 9660 CD-ROM filesystem data
 - Lastline mime type: *application/x-iso9660-image*
 - Typical extension: *.iso*
- **RarArchiveFile**: RAR archive data
 - Lastline mime type: *application/x-rar*
 - Typical extension: *.rar*
- **Rar5ArchiveFile**: RAR archive data, version 5

Lastline mime type: *application/x-rar5*

Typical extension: *.rar*

- **TarArchiveFile**: POSIX tar archive data

Lastline mime type: *application/tar*

Typical extension: *.tar*

- **XzArchiveFile**: XZ compressed data

Lastline mime type: *application/x-xz*

Typical extensions: *.xz, .txz*

- **ZipArchiveFile**: Zip archive data

Lastline mime type: *application/zip*

Typical extension: *.zip*

- **SevenZipArchiveFile**: 7-zip archive data

Lastline mime type: *application/x-7z-compressed*

Typical extension: *.7z*

- **RarSfxPeExeFile**: RAR SFX PE executable

Lastline mime type: *application/x-rar-sfx-pe*

Typical extension: *.exe*

- **ZipSfxPeExeFile**: Zip SFX PE executable

Lastline mime type: *application/x-zip-sfx-pe*

Typical extension: *.exe*

- **SevenZipSfxPeExeFile**: 7zip SFX PE executable

Lastline mime type: *application/x-7zip-sfx-pe*

Typical extension: *.exe*

NOTE: In some cases, the *Lastline mime types* shown in the above list represent a unified, generalized version of standard mime types. This allows mapping different, semantically equivalent types into a single type.

1.2 Getting Started

The Analyst API is a web-based API. To get started using it, you will need to request an *API key and API token* from Lastline. These will act as your credentials for accessing the API.

For clients accessing the API hosted in a Lastline datacenter, the API is reachable at `https://analysis.lastline.com`. For clients using an *On-Premises* deployment, the API is reachable using the URL `https://log.<fqdn>/analysis` on Lastline Enterprise Manager or Pinbox appliances, and the URL `https://<fqdn>/analysis` for Lastline Analyst appliances.

In addition to the full *API Reference*, this documentation also provides two *Sample API Clients* for accessing this API. These are written in Python. One of them is also available as a self-contained Microsoft Windows executable.

1.3 API Concepts

The Lastline Analyst API is an asynchronous API, in the sense that, when a resource (a file or a URL) is submitted for analysis, the analysis results are typically not returned immediately in the response. Instead, a unique identifier (UUID) for the submitted analysis task is returned. This UUID can later be used in a separate request to get the analysis results for this task.

The reason for this approach is that analyzing a resource can take some time. For instance, analyzing an executable requires running it for several minutes in an analysis sandbox.

However, in some cases the submitted resource may have been already analyzed by the analysis platform. In these cases, the API is able to immediately return an analysis result.

1.4 Workflow

The expected usage of this API is to follow these steps:

1. Call `submit_file()` or `submit_url()` several times to submit a number of artifacts, and store the returned task UUIDs.
2. Call `get_completed()` to get the UUIDs of tasks completed since the last time `get_completed()` was called.
3. Call `get_result()` on returned UUIDs to obtain results.
4. Repeat steps 2 and 3 until results are available for all UUIDs.

Using the `get_completed()` function avoids polling for results for each submission individually by repeatedly calling the `get_result()` function until results are available, which is very inefficient and may be enforced by the API: if a client makes too many calls to `get_result()` on incomplete tasks, it may be blocked from making further calls due to violations of this protocol.

Note that the `submit_file()` and `submit_url()` functions may immediately return an analysis result, in which case the call to `get_result()` is not necessary (the UUID will still be returned by `get_completed()`). If a client does not require the detailed analysis results at time of the submission, specify the `full_report_score` parameter. Further, the API allows submitting a file *by-hash* if the file is already available in the analysis system, avoiding an unnecessary upload of the file-content; see `submit_file()` for details.

1.5 Handling of Containers

The API analyzes submissions of archives or other containers by treating these types of files as closely as possible to how a real user would: the system tries to understand how a real victim would behave when receiving the file.

For example, when the system finds a document inside an archive, this document is sent for deeper analysis “by itself”, as documents typically are self-contained elements.

A different example is when the API handles archives with multiple programs, or when a program is shipped with additional files (such as configuration files or program libraries). In this case, it is often not meaningful to analyze each program individually, as one would expect the first program to call the other, read the configuration file embedded in the archive, or load the program library. Thus, the program would most likely fail to run successfully if analyzed “by itself” (otherwise one would not expect these files to be distributed together in the same archive).

Thus, the API may analyze this type of container containing multiple files via *program bundles*: all files in the container are copied into the analysis system, and metadata embedded in the bundle describes how to launch these files. If more

than one program is found and the system cannot identify which program to launch as “main” subject of the analysis, multiple bundle analysis runs are triggered.

For details on program bundles, see *Application Bundle Module*.

API REFERENCE

2.1 Authentication

For authentication, all Lastline Analyst API endpoints accept two methods of reading credentials: *HTTP Basic* Authentication (as defined in RFC-7617) or the following pair of authentication parameters in the request:

- `key`: Lastline Analyst API key
- `api_token`: secret Lastline Analyst API token

Since the API is accessed over https, these parameters are never sent in clear-text. Both parameters are ASCII text.

The `key` and `api_token` parameters can be embedded in each request to the Lastline Analyst API. Alternatively, the client can use the `login()` function to explicitly validate its credentials and establish an HTTP session. As long as the client provides the session token and the session is valid, the client may skip providing the credentials. When using an expired session token for authentication, the API will return the error `ANALYSIS_API_AUTHENTICATION_REQUIRED`, instructing the client to re-authenticate.

A client may mandate that the API should not establish a session by setting the HTTP header `x-nsx-lastline-no-session: 1`.

2.2 Response Format

The format of API responses can be selected by appending an extension to the request URL. Supported formats are JSON and XML. If no extension is provided, the format defaults to JSON. JSON is the recommended format for automated processing, while XML is recommended for human consumption (since modern browsers make it readable by pretty-printing it).

A successful response comes in the form:

```
{"success": 1, "data":...}
```

A failed response will return:

```
{"success": 0, "error_code": <ERROR_CODE>, "error": <ERROR_MESSAGE>}
```

The `error_code` field is optional.

Numbers returned by the API can be either decimal or hexadecimal. Hexadecimal numbers are always prefixed by '0x'.

Timestamps returned by the API are in UTC.

2.3 Methods

2.3.1 Method Index

- *submit_file()*: Submit a file (PE, PDF, doc, etc.) for analysis.
- *submit_url()*: Submit a URL for analysis (do NOT submit links to executables here!)
- *get_results()*: Fetch the results of a previously submitted analysis task
- *get_result()*: Fetch the result summary of a previously submitted analysis task
- *get_result_activities()*: Fetch information about the behavior detected as part a previously submitted analysis task
- *get_report_activities()*: Fetch information about the behavior detected as part a previously submitted analysis task, specific to a given analysis report
- *get_result_artifact()*: Fetch an artifact generated during the analysis of a previously submitted analysis task. When an artifact is associated with a specific analysis report (which is usually the case in practice), consider using *get_report_artifact()*
- *get_report_artifact()*: Fetch an artifact generated during the analysis of a previously submitted analysis task when this artifact is bound to a specific analysis report
- *get_completed()*: Get a list of analysis tasks that were recently completed
- *get_completed_with_metadata()*: Get a list of uuids, scores, and additional task metadata for recently completed analysis tasks
- *get_pending()*: Get a list of analysis tasks that are pending completion
- *get_progress()*: Get the estimated task completion for a previously submitted analysis task
- *get_task_metadata()*: Get information about a task by its UUID
- *query_file_hash()*: Query the system for results for a file using its hash
- *is_blocked_file_hash()*: Query the system for results for a file using its hash and check if the file is save for blocking using the given hash
- *query_task_artifact()*: Check if a specific task-artifact (such as the analysis subject) is available for download
- *get_network_iocs()*: Get the network IoC data for an analysis task.
- *get_ioc_metadata()*: Get information about an IOC by its UUID
- *get_ioc_report()*: Get an IOC by its UUID
- *create_ioc_from_result()*: Create an IOC based on an analysis result
- *get_api_utc_timestamp()*: Get the current UTC timestamp to be used in the API calls that require current timestamp
- *get_analysis_tags()*: Get the analysis tags for an analysis task
- *get_child_tasks_recursively()*: Get all child tasks of the given task recursively
- *export_report()*: Export a report or a combination of reports for a task
- *get_completed_exported_reports()*: Get a list of available exported analysis reports
- *get_exported_report()*: Get an exported analysis report
- *is_risky_analysis_artifact()*: Check if artifact is potentially malicious

- `login()`: Performs an initial login to service and establish a session
- `ping()`: Verifies an active session

2.3.2 Method Documentation

`malscape_service.api.views.analysis.submit_file` (*response_format*)
Submit a file for analysis.

There are two main use-cases for this function.

- Caller provides a hash of the file (through the `md5` and/or `sha1` parameters). The submit will fail if the a file with that hash is not already available to in the cloud.
- Caller uploads a file using the `file` parameter. In this case, the `md5` and `sha1` parameters are ignored.

The typical work-flow is to first attempt to submit a hash and, if that fails with error-code `ANALYSIS_API_FILE_UPLOAD_REQUIRED`, try again with the actual file.

NOTE: Submissions by file hash and submissions using the file content have identical semantics. Either submission type may trigger a new analysis, even if the file has been seen by the analysis system before. Using submissions by file hash brings the advantage that the file content may not need to be transferred, providing significantly reduced data upload for submissions of large files. The client cannot assume that a file is available for submissions by hash, even if the file has been uploaded previously (files may be deleted after analysis or may be subject to data-retention). Thus, submissions by file hash may fail with error-code `ANALYSIS_API_FILE_UPLOAD_REQUIRED` any time the file content is not provided as part of the submission.

On a successful submission, this method will return a unique identifier for the task, which can later be used to query for results using `get_results()`. Furthermore, if there are already results available for the submitted file, they will be included in the response (see function `get_results()`). If the detailed analysis report is not required, the caller may set the `full_report_score` parameter (e.g., to value -1) to suppress downloading the unnecessary data.

URL

`/analysis/submit/file[.response_format]`

response_format can be `xml` or `json` (defaults to `json`)

HTTP METHOD

POST

POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)
- **md5:** MD5 hash (in hex) of submitted file.
- **sha1:** SHA1 hash (in hex) of submitted file.
- **sha256:** SHA256 hash (in hex) of submitted file.
- **full_report_score:** Minimum score that causes detailed analysis reports to be served; -1 indicates “never return full report”; 0 indicates “return full report at all times”. (optional, default=0)
- **delete_after_analysis:** Boolean (specified as 1 or 0). If true, the backend will delete the file after analysis is done (and noone previously submitted this file with this flag set). (optional, default=0)
- **bypass_cache:** Boolean (specified as 1 or 0). If true, the API will not serve a cached result. **NOTE:** Setting this parameter requires non-default privileges. (optional, default=0)

- **analysis_timeout:** Timeout in seconds after which to terminate analysis. By default, the analysis engine automatically detect this timeout based on the behavior observed during analysis. By setting this parameter explicitly, the analysis terminates when all relevant behavior has been observed or when the given timeout is reached. NOTE: Setting this parameter requires non-default privileges. (optional, default=<depends on type of submission>)
- **analysis_env:** Enforce analysis in a specific environment. By default, the system selects the most appropriate environment(s) based on the type of file submitted and the available analysis environments. Thus, it is advisable to allow the system to auto-detect this value. The availability of different operating systems depends on the customization of the analysis system. By default (or when using the API hosted by Lastline), the following values are available:
 - *windows10*, and
 - *windows7*NOTE: Setting this parameter requires non-default privileges - any value provided in absence of the required permission will be overridden with automatically selected data. (optional, default=<depends on type of submission>)
- **allow_network_traffic:** Boolean (specified as 1 or 0). If false, all network connections will be redirected to a honeypot. NOTE: Setting this parameter requires non-default privileges. (optional, default=1)
- **filename:** Filename to use during analysis. If none is passed, the analysis engine will pick an appropriate name automatically, but submitting this information is highly recommended for obtaining best analysis results. See *Filename* for details. (optional, default=<none>)
- **keep_file_dumps:** Boolean (specified as 1 or 0). If true, more files generated during analysis will be stored for post-processing. NOTE: This can generate large volumes of data and is not recommended. NOTE: Setting this parameter requires non-default privileges. (optional, default=0)
- **keep_memory_dumps:** Boolean (specified as 1 or 0). If true, more memory snapshots taken during analysis will be kept for post-processing. NOTE: This can generate large volumes of data and is not recommended. NOTE: Setting this parameter requires non-default privileges. (optional, default=0)
- **keep_behavior_log:** Boolean (specified as 1 or 0). If true, the raw behavior log extracted during analysis will be kept for post-processing. NOTE: This can generate *very very* large volumes of data and is not recommended. NOTE: Setting this parameter requires non-default privileges. (optional, default=0)
- **push_to_portal_account:** If set, a successful submission will be pushed to the web-portal using the specified username. NOTE: It is strongly discouraged to use this parameter for automated or bulk submissions. See *Web-Portal Integration* for details. (optional, default=<none>)
- **protocol:** Protocol used to obtain the submitted file. One of “FTP”, “HTTP”, “SMB. (optional, default=HTTP)
- **server_ip:** ASCII dotted-quad representation of the IP address that was originally used to download this file. The IP is mandatory for the storage of the metadata. See *File Origin* for details. (optional, default=<none>)
- **server_port:** Integer representation of the network port number that was originally used to download this file. The port is mandatory for the storage of the metadata. See *File Origin* for details. (optional, default=<none>)
- **client_ip:** ASCII dotted-quad representation of the IP address of the client for FTP or SMB connections. (optional, default=<none>)
- **client_port:** Integer representation of the network port number of the client used for FTP or SMB connections. (optional, default=<none>)

- **direction:** One of “TO_SERVER” or “FROM_SERVER”. Use “TO_SERVER” to indicate FTP or SMB uploads, “FROM_SERVER” to indicate FTP or SMB downloads. (optional, default=“FROM_SERVER”)
- **report_version:** Version name of the report that will be returned. (optional, default=<most applicable, depends on type of report>)
- **download_ip:** *DEPRECATED*, do not use. See *server_ip*.
- **download_port:** *DEPRECATED*, do not use. See *server_port*.
- **apk_package_name:** *DEPRECATED*, do not use. Package name for APK files. (optional, default=<none>)
- **password:** Password to use when trying to analyze password-protected or encrypted content (such as archives or documents) (optional, default=<none>)
- **priority:** Priority level to set for this analysis. Priority must be between 1 and 10 (1 is the lowest priority, 10 is the highest) (optional, default=<user default>)
- **bypass_prefilter:** Boolean (specified as 1 or 0). If True, file is submitted to all supported analysis components without prior static analysis. NOTE: Setting this parameter requires non-default privileges. (optional, default=0)
- **fast_analysis:** Boolean (specified as 1 or 0). If True, file is submitted only to static analyzers NOTE: Setting this parameter requires non-default privileges. (optional, default=0)

FILE Parameters

These parameters are provided as uploaded files encoded as multipart/form-data.

- **file:** Actual body of the file to analyze
- **download_host:** HTTP host header from which this file was originally downloaded. See *File Origin* for details. (optional, default=<none>)
- **download_path:** FTP/HTTP/SMB host path from which this file was originally downloaded from or uploaded to. See *File Origin* for details. (optional, default=<none>)
- **download_user_agent:** HTTP user-agent header that was used when this file was originally downloaded. (optional, default=<none>)
- **download_referer:** HTTP refer(r)er header that was used when this file was originally downloaded. (optional, default=<none>)
- **download_request:** Full HTTP request that was originally used to download this file. (optional, default=<none>)
- **password_candidates:** A list of passwords to use when trying to analyze password-protected or encrypted content (such as archives or documents) (optional, default=<none>)

NOTE The API will reject a submission if this parameter exceeds the max number of 1000 password_candidates or if the amount of data passed is longer than 128 Kb

Error Codes

- ***ANALYSIS_API_INVALID_CREDENTIALS*:** Provided key and api_token are not valid credentials.
- ***ANALYSIS_API_PERMISSION_DENIED*:** Access to service functionality has been denied.
- ***ANALYSIS_API_FILE_UPLOAD_REQUIRED*:** When submitting by hash: file *content* required for analysis.
- ***ANALYSIS_API_FILE_TOO_LARGE*:** When submitting by file: Provided file exceeds upload limit.

NOTE: Uploaded files beyond a certain limit will be rejected by the server without reaching the Analyst API. In these cases the server will respond with *413 Request Entity Too Large*. Clients must thus handle both, *HTTP-413* and *ANALYSIS_API_FILE_TOO_LARGE*.

- ***ANALYSIS_API_TEMPORARILY_UNAVAILABLE***: Service is temporarily unavailable.
- ***ANALYSIS_API_INVALID_D_METADATA***: Invalid download metadata were specified.
- ***ANALYSIS_API_SUBMISSION_LIMIT_EXCEEDED***: The number of submissions allowed for this license has been exceeded. Certain licenses are restricted in the number or type of submissions allowed on a 24-hour interval.
- ***ANALYSIS_API_INVALID_REPORT_VERSION***: Invalid report version.
- ***ANALYSIS_API_FILE_EXTRACTION_FAILED***: Extracting files from the submitted archive failed. Either the archive format is not supported or the archive is protected with an unknown password.
- ***ANALYSIS_API_CHILD_TASK_CHAIN_TOO_DEEP***: Creating child tasks for a given analyst-engine reached the max limit
- ***ANALYSIS_API_AUTHENTICATION_REQUIRED***: Credentials or session identifier missing.
- ***ANALYSIS_API_INVALID_PRIORITY***: Invalid priority level.

Contents of successful response

- **`task_uuid`**: Unique identifier of submitted task, for use with `get_results()`
- **`submission_timestamp`**: Timestamp of when the submission was created.
- **`expires`**: The time after which the response is considered stale. The client may cache the returned task for this submission up to this time. For any submission after this time, the API may decide to return a new task.
- If results are already available, the analysis report, the score, and other additional fields are returned as in a successful request to `get_results()`.

`malscape_service.api.views.analysis.submit_url(response_format)`
Submit a URL for analysis.

On a successful submission, this method will return a unique identifier for the task, which can later be used to query for results using `get_results()`. Furthermore, if there are already results available for the submitted URL, they will be included in the response (see function `get_results()`). If the detailed analysis report is not required, the caller may set the `full_report_score` parameter (e.g., to value -1) to suppress downloading the unnecessary data.

URL

`/analysis/submit/url[.response_format]`
`response_format` can be xml or json (defaults to json)

HTTP METHOD

POST

POST Parameters

- **`key`**: Lastline API key. (required)
- **`api_token`**: Lastline API token. (required)
- **`url`**: URL to analyze. (required)
- **`referer`**: HTTP refer(r)er header to use in request (note that this is not spelled “*referrer*” to be in accordance with the original HTTP specification). (optional, default=<none>)

- **full_report_score:** Minimum score that causes detailed analysis reports to be served; -1 indicates “never return full report”; 0 indicates “return full report at all times”. (optional, default=0)
- **bypass_cache:** Boolean (specified as 1 or 0). If true, the API will not serve a cached result. NOTE: Setting this parameter requires non-default privileges. (optional, default=0)
- **analysis_timeout:** Timeout in seconds after which to terminate analysis. By default, the analysis engine automatically detect this timeout based on the behavior observed during analysis. By setting this parameter explicitly, the analysis terminates when all relevant behavior has been observed or when the given timeout is reached. NOTE: Setting this parameter requires non-default privileges. (optional, default=<depends on type of submission>)
- **push_to_portal_account:** If set, a successful submission will be pushed to the web-portal using the specified username. NOTE: It is strongly discouraged to use this parameter for automated or bulk submissions. See *Web-Portal Integration* for details. (optional, default=<none>)
- **user_agent:** HTTP User-Agent header to use in requests. (optional, default=<none>)
- **report_version:** Version name of the report that will be returned. (optional, default=<most applicable, depends on type of report>)
- **priority:** Priority level to set for this analysis. Priority must be between 1 and 10 (1 is the lowest priority, 10 is the highest) (optional, default=<user default>)
- **fast_analysis:** Boolean (specified as 1 or 0). If True, URL is submitted only to selected static analyzers. NOTE: Setting this parameter requires non-default privileges. (optional, default=0)

FILE Parameters

These parameters are provided as uploaded files encoded as multipart/form-data.

- **password_candidates:** A list of passwords to use when trying to analyze password-protected or encrypted URL content. (optional, default=<none>)

NOTE The API will reject a submission if this parameter exceeds the max number of 1000 password_candidates or if the amount of data passed is longer than 128 Kb

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and api_token are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_INVALID_URL:** Invalid URL or refer(r)er submitted.
- **ANALYSIS_API_TEMPORARILY_UNAVAILABLE:** Service is temporarily unavailable.
- **ANALYSIS_API_SUBMISSION_LIMIT_EXCEEDED:** The number of submissions allowed for this license has been exceeded. Certain licenses are restricted in the number or type of submissions allowed on a 24-hour interval.
- **ANALYSIS_API_INVALID_REPORT_VERSION:** Invalid report version.
- **ANALYSIS_API_CHILD_TASK_CHAIN_TOO_DEEP:** Creating child tasks for a given analyst-engine reached the max limit
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.
- **ANALYSIS_API_INVALID_PRIORITY:** Invalid priority level.

Contents of successful response

- **task_uuid:** Unique identifier of submitted task, for use with `get_results()`
- **submission_timestamp:** Timestamp of when the submission was created.

- **expires** The time after which the response is considered stale. The client may cache the returned task for this submission up to this time. For any submission after this time, the API may decide to return a new task.
- If results are already available, the analysis report, the score, and other additional fields are returned as in a successful request to `get_results()`.

`malscape_service.api.views.analysis.get_results(response_format)`
Get analysis results for a task.

NOTE: Do *not* call this method to query if a task has completed. This method may only be used on tasks that have been marked as completed as part of the results returned to a call of `get_completed()`. The method `get_completed()` allows to efficiently query for task completion of a large number of pending tasks in parallel without polling for the status of individual tasks. If a client violates this protocol and too frequently invokes the `get_results` method on incomplete tasks, the client may be blocked from making further calls.

URL

`/analysis/get[.response_format]`

response_format can be xml, json, RTF, or PDF (defaults to json). For RTF or PDF the result is an HTTP file download response.

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)
- **uuid:** Identifier of requested task. (required)
- **full_report_score:** Minimum score that causes detailed analysis reports to be served; -1 indicates “never return full report”; 0 indicates “return full report at all times”. If `report_uuid` is specified, this parameter is ignored. (optional, default=0)
- **report_version:** Version name of the report that will be returned. If `report_uuid` is not specified, this parameter is ignored. (optional, default=<most applicable, depends on type of report>)
- **report_uuid:** Identifier of the requested report if the task results is composed of multiple reports. Requires special permissions. (optional, default=<most applicable, depends on type of report>)
- **include_scoring_components:** Boolean (specified as 1 or 0), telling whether details of all components contributing to the overall score should be returned individually. **NOTE:** Setting this parameter requires non-default privileges. (optional, default=0)
- **allow_datacenter_redirect:** If set to False, redirection to other datacenters will be prevented. (optional, default=1)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and api_token are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_INVALID_UUID:** task_uuid is not a valid uuid (as returned by `submit_file()` or `submit_url()`).
- **ANALYSIS_API_NO_RESULT_FOUND:** No results are yet available for requested task
- **ANALYSIS_API_TEMPORARILY_UNAVAILABLE:** Service is temporarily unavailable.
- **ANALYSIS_API_INVALID_REPORT_VERSION:** Invalid report version.

- **ANALYSIS_API_NO_IOC_EXTRACTABLE:** No IOCs can be extracted from the given task (only if the requested *report_version* refers to an IOC report). See error-code description for details.
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.

Contents of successful response

- **score:** Score between 0 and 100 indicating maliciousness of the observed behavior (0=benign, 100=malicious).
- **submission:** Timestamp of when the task was created.
- **last_submission_timestamp: (optional)** Timestamp of when the task was last re-evaluated, which is the most recent submission with the current task-UUID.
- **malicious_activity: (optional)** Subset of the behavior report identifying interesting behavior. If features outside the observed behavior contribute to the maliciousness score, this field might not be available.
- **activity_to_mitre_techniques: (optional)** Dictionary that maps the malicious activities found to a list of Mitre techniques information correlated with the activity.
- **errors: (optional)** A list of error messages explaining why a submitted artifact could not be processed.
- **threat: (optional)** A threat classification.
- **threat_class: (optional)** A threat-class classification.
- **report (optional):** Analysis report for submitted resource. This field will not be returned if the *full_report_score* parameter has been provided and is greater than the score.
- **reports (optional):** Information about available analysis reports and their relevance for the analysis results.
 - **report_uuid:** Analysis report UUID (see *report_uuid* parameter).
 - **relevance:** Number (0-1) on how relevant this report is when compared to other analysis reports available for this result.
 - **description (optional):** A short description of the analysis report, such as the analysis environment.
 - **report_versions (optional):** A list of available report versions. Each report versions represents analysis information differently, focusing on different aspects of the analysis. Not all report versions apply to all reports.

For more information on the response format and report details, refer to [Analysis Results](#).

`malscape_service.api.views.analysis.get_result` (*response_format*)

Get analysis result summary for a task. This is a short version of the data returned by `get_results()`.

NOTE: Do *not* call this method to query if a task has completed. This method may only be used on tasks that have been marked as completed as part of the results returned to a call of `get_completed()`. The method `get_completed()` allows to efficiently query for task completion of a large number of pending tasks in parallel without polling for the status of individual tasks. If a client violates this protocol and too frequently invokes the `get_result` method on incomplete tasks, the client may be blocked from making further calls.

URL

`/analysis/get_result[. response_format]`

response_format can be xml or json (defaults to json)

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)
- **uuid:** Identifier of requested task. (required)
- **score_only:** If set to 1, the results only contain the score and threat/threat- class classification. Some licenses are restricted to fetching only this data. (optional, default=0)
- **allow_datacenter_redirect:** If set to False, redirection to other datacenters will be prevented. (optional, default=1)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and api_token are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_INVALID_UUID:** task_uuid is not a valid uuid (as returned by `submit_file()` or `submit_url()`).
- **ANALYSIS_API_NO_RESULT_FOUND:** No results are yet available for requested task
- **ANALYSIS_API_TEMPORARILY_UNAVAILABLE:** Service is temporarily unavailable.
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.

Contents of successful response

- See `get_results()`. Response excludes *report* format.

`malscape_service.api.views.analysis.get_result_activities(response_format)`

Get the behavior/activity information for an analysis task.

URL

`/analysis/get_result_activities[.response_format]`

response_format can be xml or json (defaults to json)

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)
- **uuid:** Identifier of requested task. (required)
- **allow_datacenter_redirect:** If set to False, redirection to other datacenters will be prevented. (optional, default=1)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and api_token are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_INVALID_UUID:** task_uuid is not a valid uuid (as returned by `submit_file()` or `submit_url()`).
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.

Contents of successful response

- **task_uuid:** The task uuid sent in the request
- **report_activities:** A list of details about each activity in the given result. Each list entry has the following keys:
 - **type:** Type of the activity (e.g. “Behavior” or “Evasion”).
 - **description:** Description of the activity.
 - **severity:** Severity of this activity.
 - **reports:** Reports in this result that contain this activity. Each list entry has the keys described below

Contents of reports entries

- **report_uuid:** Analysis report UUID (see `get_results()`).
- **has_action_ids:** If 1, the behavior is associated with specific actions (see `get_report_activities()`), otherwise 0.

`malscope_service.api.views.analysis.get_report_activities(response_format)`
Get the behavior/activity information for a specific analysis report.

URL

`/analysis/get_report_activities[.response_format]`
`response_format` can be xml or json (defaults to json)

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)
- **uuid:** Identifier of requested task. (required)
- **report_uuid:** Identifier of the report. (required)
- **allow_datacenter_redirect:** If set to False, redirection to other datacenters will be prevented. (optional, default=1)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and api_token are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_INVALID_UUID:** task_uuid is not a valid uuid (as returned by `submit_file()` or `submit_url()`).
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.

Contents of successful response

- **task_uuid:** The task uuid sent in the request
- **report_uuid:** The report uuid sent in the request
- **report_activities:** A list of details about each activity in the given report. Each list entry has the following keys:

- **type:** Type of the activity (e.g. “Behavior” or “Evasion”).
- **description:** Description of the activity.
- **severity:** Severity of this activity.
- **action_ids:** Action-IDs related to this activity (set of IDs).

`malscape_service.api.views.analysis.get_result_artifact(response_format)`
Get artifact associated with or generated by an analysis result.

NOTE: For artifacts extracted from a specific analysis report, consider using `get_report_artifact()` instead of this function.

URL

`/analysis/get_result_artifact[.response_format]`

response_format can be raw, xml, or json (defaults to raw). Files should always be downloaded in *raw* format, which serves the result as HTTP file download. Other formats are only meaningful to get a more descriptive error-code.

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)
- **uuid:** Identifier of requested task. (required)
- **artifact_uuid:** Identifier of requested task artifact in the form of <report-UUID>:<artifact-name>. (required)
- **password_protected:** If provided, use this password to protect the artifact in a zip archive. The password provided should be using only ASCII characters and have max length of 128 characters (optional)
- **allow_datacenter_redirect:** If set to False, redirection to other datacenters will be prevented. (optional, default=1)

Error Codes

Other than most API calls, this function serves raw HTTP responses if the default `response_format` (*raw*) is used. In this case, the API will use standard HTTP error codes:

- **401 - Unauthorized:** Access to service functionality has been denied.
- **404 - Not Found:** The requested artifact is not available.
- **410 - Gone:** The requested artifact is no longer available. Old analysis artifacts are cleaned up after a certain time.
- **412 - Precondition Failed:** `task_uuid` is not a valid uuid (as returned by `submit_file()` or `submit_url()`).

For `response_formats` `json` and `xml`, the API will respond using the following error status codes:

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and `api_token` are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_INVALID_UUID:** `task_uuid` is not a valid uuid (as returned by `submit_file()` or `submit_url()`).

- **`ANALYSIS_API_NO_RESULT_FOUND`**: No results are yet available for requested task
- **`ANALYSIS_API_TEMPORARILY_UNAVAILABLE`**: Service is temporarily unavailable.
- **`ANALYSIS_API_AUTHENTICATION_REQUIRED`**: Credentials or session identifier missing.
- **`:py:data: ANALYSIS_API_INVALID_ARTIFACT_UUID`**: Artifact_uuid contains an invalid artifact name and/or report uuid.

Contents of successful response

Other than most API calls, this function serves raw HTTP responses containing the artifact content if the default response_format (raw) is used.

Otherwise the response contains the following fields:

- **content**: A human-readable representation of the artifact.
- **errors: (optional)** A list of error messages explaining why a submitted artifact could not be processed.

```
malscape_service.api.views.analysis.get_report_artifact ()
```

Get artifact associated with a specific analysis report.

URL

/analysis/get_report_artifact

HTTP METHOD

GET or POST

GET/POST Parameters

- **key**: Lastline API key. (required)
- **api_token**: Lastline API token. (required)
- **uuid**: Identifier of requested task. (required)
- **report_uuid**: Identifier of the requested report to which the artifact is assigned. (required)
- **artifact_name**: Identifier of the requested assigned. (required)
- **password_protected**: If provided, use this password to protect the artifact in a zip archive. The password provided should be using only ASCII characters and have max length of 128 characters (optional)
- **allow_datacenter_redirect**: If set to False, redirection to other datacenters will be prevented. (optional, default=1)

Error Codes

Other than most API calls, this function serves raw HTTP responses. The API will use standard HTTP error codes:

- **401 - Unauthorized**: Access to service functionality has been denied.
- **404 - Not Found**: The requested artifact is not available.
- **410 - Gone**: The requested artifact is no longer available. Old analysis artifacts are cleaned up after a certain time.
- **422 - Unprocessable Entity**: The provided task, report, or artifact UUID is not valid (as returned by `get_result()`).

Contents of successful response

Other than most API calls, this function serves raw HTTP responses containing the artifact content.

`malscape_service.api.views.analysis.get_completed(response_format)`

Get the list of uuids of tasks that were completed within a given time frame. This allows to efficiently query the completion status of multiple previously submitted tasks in one call, without requiring to poll for the status for individual tasks.

The main use-case for this method is to periodically request a list of UUIDs completed since the last time this method was invoked, and then fetch each result with `get_result()`.

`get_completed_with_metadata()` should be used instead since it retrieves the score as well as any additional metadata for each task.

The format for date parameters is one of:

- **date:** 'YYYY-MM-DD'
- **datetime:** 'YYYY-MM-DD HH:MM:SS'

All times are in UTC.

URL

`/analysis/get_completed[.response_format]`

`response_format` can be xml or json (defaults to json)

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)
- **after:** Request tasks completed after this time. (required)
- **before:** Request tasks completed before this time. (optional, default=<current timestamp>)
- **include_score:** If set to 1, the result will be a mapping between completed task UUIDs and task scores. (optional, default=0)

Error Codes

- **[ANALYSIS_API_INVALID_CREDENTIALS](#):** Provided key and api_token are not valid credentials.
- **[ANALYSIS_API_PERMISSION_DENIED](#):** Access to service functionality has been denied.
- **[ANALYSIS_API_AUTHENTICATION_REQUIRED](#):** Credentials or session identifier missing.

Contents of successful response

- **after:** Start of considered time range.
- **before:** End of considered time range. If no such value was specified, the API returns the current timestamp (which can be used if the local clock is skewed)
- **resume:** The 'after' time to use for a subsequent `get_completed` call to resume collection of completed tasks.
- **more_results_available (optional):** The API limits the number of tasks returned in a single result in some situations. If this happened, this flag is set to '1' and the caller must continue fetching completed-data.
- **tasks:** List of task UUIDs of tasks completed in the specified time range either as plain list (if `include_score=0`) or as mapping between task UUIDs and scores (if `include_score=1`).

Example

To periodically request a list of completed UUIDs, the following pattern can be used:

```
ret = analysis.get_completed(after="2020-08-05 14:00:39")

# ret contents will be similar to:
#{u'data': {u'after': u'2020-08-05 14:00:39',
#           u'before': u'2020-08-07 00:25:49',
#           u'resume': u'2020-08-07 00:25:50',
#           u'tasks': [u'05243bb7443e4bba80a5890969102082',
#                     u'a8ea86b5d0d84b4bb86fedd785a9d903']}},
# u'success': 1}

# ...wait some time...

analysis.get_completed(after=ret["data"]["resume"])

# the method will respond with:
#{u'data': {u'after': u'2020-08-07 00:25:50',
#           u'before': u'2020-08-07 00:27:49',
#           u'resume': u'2020-08-07 00:27:50',
#           u'tasks': [u'e66b56f2cd4948c7ab314217e3be5cf2']}},
# u'success': 1}
```

Notes

While individual analysis reports (returned by `get_results()`) specify the start and end time of their analysis, they should be ignored for the purposes of this method. Only the values of the `after` and `before` fields in the response contents should be considered.

The `resume` parameter contains the timestamp to be used for subsequent calls to allow iterating through all available, completed analysis result UUIDs. If `more_results_available` is set then more results are immediately available.

`malscape_service.api.views.analysis.get_completed_with_metadata(response_format)`

Get the list of uuids of tasks that were completed within a given time frame along with additional metadata. This allows to efficiently query the completion status of multiple previously submitted tasks in one call, without requiring to poll for the status for individual tasks.

The main use-case for this method is to periodically request a list of UUIDs completed since the last time this method was invoked, and then fetch each result with `get_result()` and any relevant analysis errors.

The format for date parameters is one of:

- date: 'YYYY-MM-DD'
- datetime: 'YYYY-MM-DD HH:MM:SS'

All times are in UTC.

URL

`/analysis/get_completed_with_metadata[. response_format]`

`response_format` can be `xml` or `json` (defaults to `json`)

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)

- **api_token:** Lastline API token. (required)
- **after:** Request tasks completed after this time. (required)
- **before:** Request tasks completed before this time. (optional, default=<current timestamp>)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and api_token are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.

Contents of successful response

- **after:** Start of considered time range.
- **before:** End of considered time range. If no such value was specified, the API returns the current timestamp (which can be used if the local clock is skewed)
- **resume:** The 'after' time to use for a subsequent get_completed call to resume collection of completed tasks.
- **more_results_available (optional):** The API limits the number of tasks returned in a single result in some situations. If this happened, this flag is set to '1' and the caller must continue fetching completed-data.
- **tasks:** List of tasks completed in the specified time range with corresponding task UUIDs and scores with additional task metadata. The fields in each mapping are as follows:
 - **task_uuid:** The analyzed task's uuid.
 - **score:** The final score for the task after analysis is complete.
 - **insufficient_task_input_errors:** If analysis was blocked due to insufficient input to the system returns a list of error codes that describe where invalid input was provided. Error codes returned can be any of:
 - * **1001:** Failed to extract a file from an archive
 - * **1002:** Failed to decrypt an archive due to an invalid password
 - * **1003:** Failed to decrypt sample due to an invalid password

Example

To periodically request a list of completed UUIDs, the following pattern can be used:

```
ret = analysis.get_completed_with_metadata(after="2020-12-05 14:00:39")

#ret contents will be similar to:
{u'data': {u'after': u'2012-12-05 14:00:39',
          u'before': u'2012-12-07 00:25:49',
          u'tasks': [
            {
              u'task_uuid': u'05243bb7443e4bba80a5890969102082',
              u'score': 50,
              u'insufficient_task_input_errors': [1001]
            },
            {
              u'task_uuid': u'a8ea86b5d0d84b4bb86fedd785a9d903',
              u'score': 29
            }
          ]
        }
u'success': 1}
```

```
# ...wait some time...

analysis.get_completed_with_metadata(after=ret["data"]["before"])

# the method will respond with:
{u'data': {u'after': u'2020-12-07 00:25:49',
          u'before': u'2020-12-07 00:27:49',
          u'tasks': [
            {
              u'task_uuid': u'e66b56f2cd4948c7ab314217e3be5cf2',
              u'score': 50,
              u'insufficient_task_input_errors': [1004]
            }
          ]
        }
u'success': 1}
```

Notes

While individual analysis reports (returned by `get_results()`) specify the start and end time of their analysis, they should be ignored for the purposes of this method. Only the values of the after and before fields in the response contents should be considered.

If `more_results_available` is set, the `before` parameter contains the timestamp to be used for subsequent calls to allow iterating through all available, completed analysis result UUIDs.

`malscape_service.api.views.analysis.get_pending(response_format)`

Get the list of uuids of pending tasks that were created within a given time frame.

The format for date parameters is one of:

- date: 'YYYY-MM-DD'
- datetime: 'YYYY-MM-DD HH:MM:SS'

All times are in UTC.

URL

`/analysis/get_pending[. response_format]`

`response_format` can be xml or json (defaults to json)

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)
- **after:** Request pending tasks created after this time. (optional, default=<since epoch>)
- **before:** Request pending tasks created before this time. (optional, default=<current timestamp>)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and api_token are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.

Contents of successful response

- **after:** Start of considered time range. If no such value was specified, the API returns the time created for the first submission returned (or current timestamp if no submissions returned).
- **before:** End of considered time range. If no such value was specified, the API returns the current timestamp (which can be used if the local clock is skewed)
- **more_results_available (optional):** The API limits the number of tasks returned in a single result in some situations. If this happened, this flag is set to '1' and the caller must continue fetching completed-data.
- **tasks:** List of task UUIDs of tasks completed in the specified time range as plain list.

Example

To periodically request a list of pending UUIDs, the following pattern can be used:

```
ret = analysis.get_pending()

# ret contents will be similar to:
#{u'data': {u'after': u'2020-08-05 14:00:39',
#           u'before': u'2020-08-07 00:25:49',
#           u'tasks': [u'05243bb7443e4bba80a5890969102082',
#                     u'a8ea86b5d0d84b4bb86fedd785a9d903']}},
# u'success': 1}

# ...wait some time...

analysis.get_pending(after=ret["data"]["resume"])

# the method will respond with:
#{u'data': {u'after': u'2020-08-07 00:25:49',
#           u'before': u'2020-08-07 00:27:49',
#           u'tasks': [u'e66b56f2cd4948c7ab314217e3be5cf2']}},
# u'success': 1}
```

Notes

The *resume* parameter contains the timestamp to be used for subsequent calls to allow iterating through all available, pending analysis result UUIDs. If *more_results_available* is set then more results are immediately available.

`malscape_service.api.views.analysis.get_progress(response_format)`

Get analysis progress for a task.

NOTE: Do *not* call this method to query if a task has completed or not. This method estimates the analysis progress on a scale of 0 to 100, thus providing more data than the client may need. The method `get_completed()` allows to efficiently query for task completion of a large number of pending tasks in parallel without polling for the status of individual tasks. If a client violates this protocol and too frequently invokes the `get_progress` method, the client may be blocked from making further calls.

URL

`/analysis/get_progress[. response_format]`

response_format can be xml or json (defaults to json)

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)

- **uuid:** Identifier of requested task. (required)
- **allow_datacenter_redirect:** If set to False, redirection to other datacenters will be prevented. (optional, default=1)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and api_token are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_INVALID_UUID:** task_uuid is not a valid uuid (as returned by `submit_file()` or `submit_url()`).
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.

Contents of successful response

- **progress:** Value between 0 and 100 indicating completion of the analysis.
- **completed:** 1 if the sample has completed, otherwise 0.

`malscape_service.api.views.analysis.get_task_metadata(response_format)`
Get information about a task by its UUID.

URL

`/analysis/get_task_metadata[. response_format]`

`response_format` can be xml or json (defaults to json)

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required for some license types)
- **uuid:** Identifier of requested task. (required)
- **allow_datacenter_redirect:** If set to False, redirection to other datacenters will be prevented. (optional, default=1)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and api_token are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_INVALID_UUID:** uuid is not a valid uuid (as returned by `submit_file()` or `get_results()`).
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.

Contents of successful response

The available details highly depend on the type of task. Available data includes:

- **task_uuid: (required)** Identifier of the task.
- **task_type: (optional)** 'file' for tasks generated from a file submission, 'url' for tasks generated from a URL submission.
- **file_md5: (optional)** MD5 hash of the file for which the task was generated, if it was generated from a file submission.

- **file_sha1: (optional)** SHA1 hash of the file for which the task was generated, if it was generated from a file submission.
- **file_sha256: (optional)** SHA256 hash of the file for which the task was generated, if it was generated from a file submission.
- **file_mime_type: (optional)** Mime-type of the file for which the task was generated, if it was generated from a file submission.
- **filename: (optional)** Name of the file that was sent during the file submission, if it was provided during the submission.
- **url: (optional)** URL for which the task was generated, if it was generated from a URL submission.
- **referrer: (optional)** Referrer used for analysis, if the task was generated from a URL submission and a referer was specified.

`malscape_service.api.views.analysis.query_file_hash(response_format)`

Query if the file has been seen before.

NOTE: Other than `submit_file()`, this function will always return a completed analysis result, if one is available. In contrast, `submit_file()` will verify if the submission should be reanalyzed to guarantee best results. Reasons for reanalysis include the availability of new/updated detectors, updated meta-information, or analysis report expiration.

URL

`/analysis/query/file_hash[. response_format]`

`response_format` can be `xml` or `json` (defaults to `json`)

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)
- **hash_algorithm:** Hash algorithm used in the query. One of 'MD5', 'SHA1', or 'SHA256'. (required)
- **hash_block_size:** Size of the block in bytes (beginning at the start of the file) on which the hash was computed. NOTE: Do not submit the file-size as block-size to query for a hash on the entire file. Instead, omit the parameter. (optional, default=<full file size>)
- **hash_value:** Value (in hex) of the hash (using 'hash-algorithm' on the first 'hash-block-size' bytes, required)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and `api_token` are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_INVALID_HASH_ALGORITHM.** An invalid hash-algorithm has been provided.
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.

Contents of successful response

- **files_found:** Number of files that matched the query.
- **tasks: (optional)** List of analysis tasks matching the query.

- **file_md5: (optional)** MD5 of the (complete) file matching the query.
- **file_sha1: (optional)** SHA1 of the (complete) file matching the query.
- **file_sha256: (optional)** SHA256 of the (complete) file matching the query.
- **task_uuid:** Task UUID to the most recent analysis task of the file matching the query.
- **score:** Score between 0 and 100 indicating maliciousness of the observed behavior (0=benign, 100=malicious).
- **insufficient_task_input_errors:** If analysis was blocked due to insufficient input to the system, returns a list of error codes that describe where invalid input was provided.
- **expires:** The time after which the response is considered stale. The client may cache the returned task for this submission up to this time. For any submission after this time, the API may decide to return a new task.

`malscape_service.api.views.analysis.is_blocked_file_hash(response_format)`

Query if the file has been seen before and enough information has been gathered to consider the file as blockable.

URL

`/analysis/query/is_blocked_file_hash[. response_format]`

response_format can be xml or json (defaults to json)

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required for some license types)
- **hash_algorithm:** Hash algorithm used in the query. One of 'MD5', 'SHA1', or 'SHA256'. (required)
- **hash_block_size:** Size of the block in bytes (beginning at the start of the file) on which the hash was computed. NOTE: Do not submit the file-size as block-size to query for a hash on the entire file. Instead, omit the parameter. (optional, default=<full file size>)
- **hash_value:** Value (in hex) of the hash (using 'hash-algorithm' on the first 'hash-block-size' bytes). (required)

Error Codes

- **[ANALYSIS_API_INVALID_CREDENTIALS](#):** Provided key and api_token are not valid credentials.
- **[ANALYSIS_API_PERMISSION_DENIED](#):** Access to service functionality has been denied.
- **[ANALYSIS_API_INVALID_HASH_ALGORITHM](#).** An invalid hash-algorithm has been provided.
- **[ANALYSIS_API_AUTHENTICATION_REQUIRED](#):** Credentials or session identifier missing.

Contents of successful response

- **decision:**

One of

- **FILE_UNKNOWN:** The file is not known,
- **BLOCK:** The client should assume the file to be malicious,
- **NOBLOCK:** The client should not assume the file to be malicious.

NOTE: This does not indicate that the file is benign, it merely means that using the given hash, there is no (or not enough) evidence to make a decision.

- **file_md5: (optional)** MD5 of the (complete) file matching the query if the file is known.
- **file_sha1: (optional)** SHA1 of the (complete) file matching the query if the file is known.
- **file_sha256: (optional)** SHA256 of the (complete) file matching the query if the file is known.

`malscape_service.api.views.analysis.query_task_artifact` (*response_format*)

Query if a specific task artifact is available for download.

URL

`/analysis/query_task_artifact[. response_format]`

response_format can be **raw, xml, or json (defaults to raw)**. For raw, the result is an HTTP file download response.

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)
- **uuid:** Identifier of requested task. (required)
- **artifact_name:** Identifier of task artifact. Most tasks allow to download the file that was submitted for analysis - to query for this artifact, use `artifact_name` *primary_analysis_subject*. (required)
- **allow_datacenter_redirect:** If set to False, redirection to other datacenters will be prevented. (optional, default=1)

Error Codes

- ***ANALYSIS_API_INVALID_CREDENTIALS*:** Provided key and api_token are not valid credentials.
- ***ANALYSIS_API_PERMISSION_DENIED*:** Access to service functionality has been denied.
- ***ANALYSIS_API_INVALID_UUID*:** task_uuid is not a valid uuid (as returned by `submit_file()` or `submit_url()`).
- ***ANALYSIS_API_AUTHENTICATION_REQUIRED*:** Credentials or session identifier missing.

Contents of successful response

- **available:** 1 if the artifact is available, otherwise 0.
- **task_uuid: (optional)** Task UUID (see `get_result_artifact()`) for which the artifact can be downloaded (if *available* is set to 1).
- **report_uuid: (optional)** Analysis report UUID (see `get_result_artifact()` function) for which the artifact can be downloaded (if *available* is set to 1).
- **artifact_name: (optional)** Name under which the artifact can be downloaded (see `get_result_artifact()` function, if *available* is set to 1).

`malscape_service.api.views.analysis.get_network_iocs` (*response_format*)

Get the network IoC data for an analysis task.

URL

`/analysis/ioc/get_network_iocs[. response_format]`
response_format can be xml or json (defaults to json)

HTTP METHOD

GET

GET Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)
- **uuid:** Identifier of requested task. (required)
- **allow_datacenter_redirect:** If set to False, redirection to other datacenters will be prevented. (optional, default=1)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and api_token are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_INVALID_UUID:** task_uuid is not a valid uuid (as returned by `submit_file()` or `submit_url()`).
- **ANALYSIS_API_NO_RESULT_FOUND:** No results are yet available for requested task
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.

Contents of successful response

- **task_uuid:** The task uuid sent in the request
- **network_iocs:** A list of network iocs extracted from pcaps. Each list entry has the following keys:
 - **report_uuid:** Analysis report that generated this PCAP data
 - **pcap_info_version:** The version number of the pcap data
 - **pcap_info:** The pcap data with the network ioc information. For more information, please look at the explanation under *Contents of Pcap Info*

Contents of Pcap Info

- **ioc_list** List of information on specific IOCs
 - **ioc_type** Type of IoC being reported (host|IP|...)
 - **value** Value of the associated IoC
 - **tags** List of string tags on the specific IoC value

`malscape_service.api.views.analysis.get_ioc_metadata(response_format)`

Get information about an IOC by its UUID.

URL

`/analysis/ioc/get_ioc_metadata[. response_format]`
response_format can be xml or json (defaults to json)

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required for some license types)
- **ioc_uuid:** Identifier of IOC. (required)
- **allow_datacenter_redirect:** If set to False, redirection to other datacenters will be prevented. (optional, default=1)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and api_token are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_INVALID_UUID:** ioc_uuid is not a valid uuid (as returned by `create_ioc_from_result()` or `get_results()`).
- **ANALYSIS_API_NO_RESULT_FOUND:** No results are yet available for requested IOC.
- **ANALYSIS_API_TEMPORARILY_UNAVAILABLE:** Service is temporarily unavailable.
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.

Contents of successful response

The available details highly depend on the type of IOC or the origin of the IOC (which system triggered its generation). Available data includes:

- **ioc_uuid: (required)** Identifier of the IOC.
- **task_uuid: (optional)** Identifier of task for which the IOC was generated, if the IOC was generated from a single analysis task.
- **report_uuid: (optional)** Identifier of the report the IOC was generated for, if the IOC was generated from a single analysis report.
- **report_version: (optional)** Version name of the IOC report that this IOC UUID is referring to.
- **task_score: (optional)** Score associated with the analysis task, if the IOC was generated from a single analysis task, as returned by `get_results()`.
- **file_md5: (optional)** MD5 hash of the file for which the IOC was extracted, if it was generated from a single file.
- **file_sha1: (optional)** SHA1 hash of the file for which the IOC was extracted, if it was generated from a single file.
- **file_mime_type: (optional)** Mime-type of the file for which the IOC was extracted, if it was generated from a single file.
- **report_env: (optional)** Analysis environment description (such as the operating system) on which data for the IOC was extracted, if it was extracted from a single analysis report.

`malscape_service.api.views.analysis.get_ioc_report(response_format)`

Get an IOC by its UUID.

NOTE: This function does not define the format in which the results are returned, as this is mandated by the IOC-UUID already.

URL

`/ioc/get_ioc_report`

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required for some license types)
- **ioc_uuid:** Identifier of IOC. (required)

Error Codes

See function `get_results()`

Contents of successful response

See function `get_results()`

`malscape_service.api.views.analysis.create_ioc_from_result(response_format)`

Create an IOC based on an analysis result.

URL

`/analysis/ioc/create_ioc_from_result[.response_format]`

`response_format` can be xml or json (defaults to json).

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)
- **uuid:** Identifier of requested task. (required)
- **report_version:** Type of IOC report to create. (required)
- **report_uuid:** Identifier of the requested report. (required)
- **allow_datacenter_redirect:** If set to False, redirection to other datacenters will be prevented. (optional, default=1)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and api_token are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_INVALID_UUID:** task_uuid is not a valid uuid (as returned by `submit_file()` or `submit_url()`).
- **ANALYSIS_API_NO_RESULT_FOUND:** No results are available for requested task.
- **ANALYSIS_API_TEMPORARILY_UNAVAILABLE:** Service is temporarily unavailable.
- **ANALYSIS_API_INVALID_REPORT_VERSION:** Invalid report version.
- **ANALYSIS_API_NO_IOC_EXTRACTABLE:** No IOCs can be extracted from the given task. See error-code description for details.
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.

Contents of successful response

- See `get_ioc_metadata()`.

`malscape_service.api.views.analysis.get_api_utc_timestamp(response_format)`

Get the current UTC timestamp to be used in the API on calls that can require the current timestamp, such as `get_completed` for example

URL

`/analysis/get_api_utc_timestamp[.response_format]`

`response_format` can be `xml` or `json` (defaults to `json`)

HTTP METHOD

GET

GET Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and `api_token` are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.

Contents of successful response

- **timestamp:** The current timestamp for the API

`malscape_service.api.views.analysis.get_analysis_tags(response_format)`

Get the analysis tags for an analysis task.

URL

`/analysis/get_analysis_tags[.response_format]`

`response_format` can be `xml` or `json` (defaults to `json`)

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)
- **uuid:** Identifier of requested task. (required)
- **allow_datacenter_redirect:** If set to `False`, redirection to other datacenters will be prevented. (optional, default=1)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and `api_token` are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_INVALID_UUID:** `task_uuid` is not a valid uuid (as returned by `submit_file()` or `submit_url()`).
- **ANALYSIS_API_NO_RESULT_FOUND:** No results are yet available for requested task
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.

Contents of successful response

- **task_uuid:** The task uuid sent in the request
- **analysis_tags:** A list of details about each tag in the given result. Each list entry has the following keys:
 - **format:** Format of the tag.
 - **data:** Content of the tag data and score
 - * **type** : source of the tag. **format** = typed_tag , **value** : tag value.

`malscape_service.api.views.analysis.get_child_tasks_recursively(response_format)`
Get a list of child tasks of the given task UUID recursively

URL

`/analysis/get_child_tasks_recursively[.response_format]`

`response_format` can be xml or json (defaults to json)

HTTP METHOD

GET or POST

GET/POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required for some license types)
- **uuid:** Identifier of task. (required)
- **allow_datacenter_redirect:** If set to False, redirection to other datacenters will be prevented. (optional, default=1)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and api_token are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_INVALID_UUID:** uuid is not a valid uuid (as returned by `submit_file()` or `get_results()`).
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.

Contents of successful response

- **task_uuid:** The task uuid sent in the request
- **child_tasks:** A mapping between the child task UUID and child task information
 - **depth:** The depth of the child task. For example, if the depth is 1, it means the child task is the direct child task of the submitted task UUID. If the depth is 2, it means the child task is the grandchild task (the child task of the child task) of the submitted task UUID

`malscape_service.api.views.analysis.export_report(response_format)`
Export a report or a combination of reports for a task.

Note: This method is currently available only to customers using the hosted, software-as-a-service API deployment.

This method allows exporting analysis results. This export works asynchronously, and a call to this method triggers the generation of a report. On a successful request, this method will return a unique identifier for the report along with an event id that can be used to get report completion status using the API call `get_completed_exported_reports()`. Once a report identifier has successfully been returned by `get_completed_exported_reports()`, the report itself can be obtained with the API call `get_exported_report()`.

URL

`/analysis/export_report[. response_format]`

response_format can be xml or json (defaults to json)

HTTP METHOD

POST

POST Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)
- **uuid:** Identifier of requested task. (required)
- **report_type:** The kind of report to export. (required)
 - Supported types:
 - * **OVERVIEW:** The exported report contains an overview of the analysis.
 - * **ALL:** The exported report contains an overview of the analysis and all reports for the task.
 - * **FULL:** The exported report contains an overview of the analysis and all reports for the task and all child tasks (recursively).
- **report_format:** The format of the generated report. (optional, default="PDF")
 - Supported formats:
 - * **PDF**

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and api_token are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.
- **ANALYSIS_API_INVALID_UUID:** task_uuid is not a valid uuid (as returned by `submit_file()` or `submit_url()`).
- **ANALYSIS_API_NO_RESULT_FOUND:** No results are yet available for requested task
- **ANALYSIS_API_TEMPORARILY_UNAVAILABLE:** Service is temporarily unavailable.

Contents of successful response

- **exported_report_uuid:** Unique identifier for the report.
- **resume_after_report_uuid:** An event id for use with `get_completed_exported_reports()`. If omitted, then omit the value when calling `get_completed_exported_reports()`.

`malscape_service.api.views.analysis.get_completed_exported_reports(response_format)`
Get a list of exported reports that are available for download.

Note: This method is currently available only to customers using the hosted, software-as-a-service API deployment.

URL

`/analysis/get_completed_exported_reports[. response_format]`

response_format can be xml or json (defaults to json)

HTTP METHOD

GET

GET Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required for some license types)
- **resume_after_report_uuid:** The last consumed report UUID. To enumerate reports ready for download, specify the last UUID that the client processed; the API will return UUIDs of new reports that were made available after the given one. If not provided, will return all stored reports. If the API returns no reports then use the same *resume_after_report_uuid* for the following call to the API. (optional)

Error Codes

- **ANALYSIS_API_INVALID_CREDENTIALS:** Provided key and api_token are not valid credentials.
- **ANALYSIS_API_PERMISSION_DENIED:** Access to service functionality has been denied.
- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.
- **ANALYSIS_API_INVALID_UUID:** The UUID provided in resume_after_report_uuid is invalid.

Contents of successful response

- **available_reports:** A list of reports with the following info, ordered by report completion time. - **exported_report_uuid:**
 - The UUID of the report
 - **task_uuid:** UUID of the task associated with this report
 - **export_timestamp:** Timestamp of the creation of the exported report.
 - **export_error: (optional)** An error message explaining why a report could not be exported.

```
malscape_service.api.views.analysis.get_exported_report ()
```

Get a completed exported report.

Note: This method is currently available only to customers using the hosted, software-as-a-service API deployment.

URL

/analysis/get_exported_report

HTTP METHOD

GET

GET Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required for some license types)
- **exported_report_uuid:** The uuid of the report to receive (required).
- **cdn:** Boolean (specified as 1 or 0). If True, may redirect to CDN to retrieve the exported report. If False, will directly return the exported report (optional, default is 1).

Error Codes

Other than most API calls, this function serves raw HTTP responses. The API will use standard HTTP error codes:

- **401 - Unauthorized:** Access to service functionality has been denied.

- **404 - Not Found:** The requested artifact is not available.
- **422 - Unprocessable Entity:** The provided report UUID is not valid.

Contents of successful response

Other than most API calls, this function serves raw HTTP responses containing the exported report content.

`malscape_service.api.views.analysis.is_risky_analysis_artifact` (*response_format*)
Determine if artifact associated with or generated by an analysis result could be malicious.

URL

`/analysis/is_risky_analysis_artifact[. response_format]`

response_format can be xml or json (defaults to json)

HTTP METHOD

GET

GET Parameters

- **key:** Lastline API key. (required)
- **api_token:** Lastline API token. (required)
- **artifact_uuid:** Identifier of requested task artifact in the form of <report-UUID>:<artifact-name>. (required)
- **uuid:** Identifier of requested task used to redirect to the correct datacenter. (optional)
- **allow_datacenter_redirect:** If set to False, redirection to other datacenters will be prevented. (optional, default=1)

Error Codes

- **[ANALYSIS_API_INVALID_ARTIFACT_UUID](#):** Provided artifact_uuid contains an invalid artifact name and/or report uuid.
- **[ANALYSIS_API_TEMPORARILY_UNAVAILABLE](#):** Service is temporarily unavailable.

Contents of successful response

- **is_risky:** '1' if the artifact could be malicious, '0' if safe.

`malscape_service.api.views.authentication.login` (*response_format*)
Authenticate with the API.

URL

`/authentication/login[. response_format]`

response_format can be xml or json (defaults to json)

HTTP METHOD

POST

POST Parameters

- **key:** Lastline license key.
- **api_token:** Lastline API-token. (optional, but required for many APIs)

Contents of successful response

“success”

Error Codes

- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.
- **ANALYSIS_API_INVALID_CREDENTIALS:** Invalid pair of ‘key’/‘api_token’ were provided.
- **ANALYSIS_API_PERMISSION_DENIED:** Credentials are valid, but current user have no rights to call this API.

`malscape_service.api.views.authentication.ping(response_format)`
Validate authenticated session.

URL

`/authentication/ping[.response_format]`

response_format can be xml or json (defaults to json)

HTTP METHOD

GET or POST

Contents of successful response

“pong”

Error Codes

- **ANALYSIS_API_AUTHENTICATION_REQUIRED:** Credentials or session identifier missing.
- **ANALYSIS_API_INVALID_CREDENTIALS:** Invalid pair of ‘key’/‘api_token’ were provided.
- **ANALYSIS_API_PERMISSION_DENIED:** Credentials are valid, but current user have no rights to call this API.

2.4 Error Codes

`malscape_service.api.views.analysis.ANALYSIS_API_FILE_UPLOAD_REQUIRED`
Error code 101: Returned when submitting a file using the file’s hash, but the file *content* is required for performing an analysis. Submit the actual file instead - see `submit_file()` for details.

Note: Even if the file content has been uploaded for analysis previously, it is possible that this error is returned.

Note: The optimized *Workflow* of submitting files for analysis suggests submitting a file by the file’s MD5 and SHA1 hashes first. This can avoid transmitting large files when an upload may not be required. If this error is returned and the client repeats the submission to upload the file content, the first submission is not counted against a client’s submission quota.

`malscape_service.api.views.analysis.ANALYSIS_API_FILE_NOT_AVAILABLE`
See `ANALYSIS_API_FILE_UPLOAD_REQUIRED`

`malscape_service.api.views.analysis.ANALYSIS_API_INVALID_CREDENTIALS`
Error code 104: No valid API credentials were provided. All requests to the Lastline Analyst API require a valid API key and API token.

`malscape_service.api.views.analysis.ANALYSIS_API_INVALID_UUID`

Error code 105: Returned when requesting results for a task, if the provided UUID is not valid. Make sure that the UUID parameter is being provided as returned by `submit_file()` or `submit_url()`

`malscape_service.api.views.analysis.ANALYSIS_API_NO_RESULT_FOUND`

Error code 106: Returned when requesting results for a task, if the results are not yet available. Please try again later, or use the `get_completed()` API function to obtain a list of tasks that have already completed.

`malscape_service.api.views.analysis.ANALYSIS_API_TEMPORARILY_UNAVAILABLE`

Error code 107: API functionality is temporarily unavailable. Please try again later.

`malscape_service.api.views.analysis.ANALYSIS_API_PERMISSION_DENIED`

Error code 108: Access to service functionality has been denied.

`malscape_service.api.views.analysis.ANALYSIS_API_FILE_TOO_LARGE`

Error code 109: When submitting by file: Provided file exceeds upload limit. In the default configuration, the upload-limit is 10MB.

`malscape_service.api.views.analysis.ANALYSIS_API_INVALID_D_METADATA`

Error code 112: Invalid download metadata was submitted.

`malscape_service.api.views.analysis.ANALYSIS_API_INVALID_FILE_TYPE`

Error code 113: When submitting by file: Provided file-type is not supported by any available analysis engine.

`malscape_service.api.views.analysis.ANALYSIS_API_INVALID_ARTIFACT_UUID`

Error code 114: The requested artifact UUID was invalid or the requested artifact is no longer available (e.g., data retention policies caused it to be deleted).

`malscape_service.api.views.analysis.ANALYSIS_API_SUBMISSION_LIMIT_EXCEEDED`

Error code 115: The license used during a submission is limited to a certain maximum number of submissions per day. This limit has been exceeded.

`malscape_service.api.views.analysis.ANALYSIS_API_INVALID_HASH_ALGORITHM`

Error code 116: The given algorithm is either invalid or unknown.

`malscape_service.api.views.analysis.ANALYSIS_API_INVALID_URL`

Error code 117: An invalid URL or refer(r)er was submitted.

`malscape_service.api.views.analysis.ANALYSIS_API_INVALID_REPORT_VERSION`

Error code 118: The requested report version is invalid.

`malscape_service.api.views.analysis.ANALYSIS_API_FILE_EXTRACTION_FAILED`

Error code 119: The submitted file is an archive and the API was unable to extract the contained files. This can be due to an incorrectly specified decryption password in `submit_file()`.

`malscape_service.api.views.analysis.ANALYSIS_API_NO_IOC_EXTRACTABLE`

Error code 120: The requested report cannot be expressed as an IOC in the requested format. Typically this happens when the analysis report does not contain sufficient actions to represent the behavior in a meaningful IOC report, or the set of actions exported as IOC report would cause too many false positives. It is also possible that the requested IOC report format does not allow expressing the observed behavior accurately, for example when an observable cannot be described in the chosen IOC format.

`malscape_service.api.views.analysis.ANALYSIS_API_CHILD_TASK_CHAIN_TOO_DEEP`

Error code 121: Internal, do not use. The submission was rejected due to generation of overly long chains of analysis child tasks.

`malscape_service.api.views.analysis.ANALYSIS_API_AUTHENTICATION_REQUIRED`

Error code 122: The request does not contain the required credentials and no valid session identifier was provided (or the session has expired).

`malscape_service.api.views.analysis.ANALYSIS_API_DATA_NO_LONGER_AVAILABLE`

Error code 123: The requested data is no longer available (e.g., data retention policies caused it to be deleted).

2.5 Submission Metadata

While many of the parameters for `submit_file()` and `submit_url()` are technically optional, it is highly recommended to provide these values to guarantee the most accurate analysis results.

2.5.1 Filename

The Lastline analysis system (including the API) uses the content of an uploaded file to determine the file's type. However, for some files, the file content can be ambiguous and the intended way to analyze a file cannot be determined reliably from the content alone.

Consider the case of examining encrypted files, such as archives, PDFs, or Microsoft Office documents: much of the actual content used for determining the type is not available - either because a password or passphrase is required to access the content, or because the file format is proprietary. As a result, the file type detection needs to incorporate the filename in the decision logic to determine how (and if) the file should be analyzed.

Or consider the following example:

```
function calculate(value1, value2) {  
    return value1 + value2;  
}
```

By looking at the file content alone, it is not straight-forward to decide what programming or scripting language this is. Therefore, it is not clear how to run, open, or analyze the given file:

- it could be a JavaScript file meant for running in a browser, or
- it could be a JavaScript file to be executed via JScript.exe on a Microsoft Windows host, or
- it could be Powershell script for Microsoft Windows, or
- it could be C or C++ depending on the context in which it is compiled, or
- it could simply be a text file that resembles code.

The analysis system attempts to find the best fit for a given file, and if a file could be interpreted differently, the file is executed in multiple ways. If there are too many options and the system cannot find a reasonable set of candidates, the file may be rejected. By using the filename in addition to the file content, the analysis system can make a better decision and more accurately analyze the file.

Providing the filename is not only important for file-type detection, it may also affect how a file behaves: The Microsoft Windows operating system, for example, may execute programs differently depending on the name of the file, as the OS makes the UI and system interact with the user differently if the filename suggests that the program is an installer (just as one example).

2.5.2 File Origin

Information on the origin of a file, such as provided via `download_host` or `download_path` may improve the classification accuracy. By using such metadata, the system can track reputation information and can identify reputable programs more reliably.

2.6 Web-Portal Integration

The `submit_file()` and `submit_url()` allow specifying an optional `push_to_portal_account` parameter for forwarding successful API submissions to the web-portal *Submission history* tab.

This *Submission history* tab is not designed to store large numbers of submissions. As a result, **it is strongly discouraged to use/set the parameter** when doing automated or bulk submissions using this API. Populating the *Submission history* with excessive data can negatively impact the user experience of the web-portal. The integration with the portal is currently considered for deprecation in a future version.

ANALYSIS RESULTS

The Analyst API returns analysis results via function `get_results()` and a `task_uuid` returned by a previous call to `submit_file()` or `submit_url()`.

Every response from `get_results()` contains a number of mandatory fields providing submission information, analysis results, and a classification of the submitted artifact.

Additionally, each response contains optional information, depending on the type of submission (file or URL submission), analysis outcome, and analysis verbosity settings.

Contents of response

- **task_uuid.** *Type:* Hexadecimal string.

Example: 7065a3ba0c729ad5981a1e1072df710d.

Unique identifier for this analysis submission.

- **score.** *Type:* Integer.

Example: 75.

Score between 0 and 100 indicating maliciousness of the observed behavior (0=benign, 100=malicious). This allows the client to choose which artifacts to consider malicious according to its own security policies. Lastline standard/best-practice is to consider any artifact with

- score < 30 to be benign,
- score >= 30 and score < 70 to be suspicious / a nuisance, and
- score >= 70 to be malicious.

- **submission.** *Type:* Date-Time.

Example: 2013-10-09 12:12:12.

Timestamp of the submission that triggered the initial analysis.

- **last_submission_timestamp (optional).** *Type:* Date-Time.

Example: 2013-10-09 12:12:12.

Timestamp of the most recent submission that returned this analysis task.

- **analysis_subject.** *Type:* Dictionary.

- **url: (optional).** *Type:* String.

Example: “http://www.example.com/”.

URL submitted for analysis (URL analysis only).

- **referer: (optional).** *Type:* String.
Example: “http://www.referer.com/”.
Referer/Referrer for analysis (URL analysis only).
- **md5: (optional).** *Type:* String.
Example: “c9d2242bbaca823b80916fec27e9f2bb”.
File MD5 hash (file analysis only).
- **sha1: (optional).** *Type:* String.
Example: “c9d2242bb263483837456fec27e9f2bb”.
File SHA1 hash (file analysis only).
- **mime_type: (optional).** *Type:* String.
Example: “application/pdf”.
File mime-type (file analysis only).

Additional response contents for successful analysis

- **malicious_activity. (optional).** *Type:* List of strings.
Example: “Type1: Value1”, “Type2: Value2”.
Subset of the behavior report identifying malicious behavior. If features outside the observed behavior contribute to the maliciousness score or no malicious behavior was detected during analysis, this field will not be available.
- **threat. (optional).** *Type:* String.
Example: “Zeus”.
A threat classification. If the analysis subject was classified as benign or does not belong to a known threat family, this field will not be available.
- **threat_class. (optional).** *Type:* String.
Example: “Command & Control”.
A threat-class classification. If the analysis subject was classified as benign or does not belong to a known threat class family, this field will not be available.
- **report. (optional).** *Type:* See *Analysis Report Format*.
Analysis report for submitted resource. If an analysis subjects fails be analyzed but can still be classified using other information, the response will not contain this field.
- **child_tasks. (optional).** *Type:* List of child tasks. See *Child Tasks*.
- **reports. (optional).** *Type:* List of dictionaries.
 - **report_uuid:** *Type:* See *report uuid*.
Analysis report UUID.
 - **relevance:** *Type:* Integer.
Example: 0.
Number (0-1) on how relevant this report is when compared to other analysis reports available for this result.

- **description (optional):** *Type:* String.

Example: “Dynamic analysis on Microsoft Windows 7”.

A short description of the analysis report, such as the analysis environment. For details, see *Report Descriptions*.

- **report_versions (optional):** *Type:* List of Strings.

Example: ‘ll-int-win’, ‘ll-win-timeline-based’.

A list of available report versions. Each report version represents analysis information differently, focusing on different aspects of the analysis. Not all report versions apply to all reports. For a list of report versions, see *report versions*.

Additional response contents for failed analysis

- **errors.** *Type:* List of strings.

Example: “Error1”, “Error2”.

A list of error messages explaining why a submitted artifact could not be processed.

3.1 Analysis Report Format

The Analyst API uses different internal analysis engines to analyze a submission of a supported type. The analysis engine determines the fields in the report and the report’s format.

Report contents

- **uuid.** *Type:* Hexadecimal string.

Example: 2fbffe68406f500f3e3ef9c:ba675cc0ey-qSdKdW1_rEA.

Unique identifier for the analysis report. This value can be used to retrieve result artifacts or analysis metadata (see *get_result_artifact()*).

- **format.** *Type:* Dictionary.

- **name.** *Type:* String.

Example: “ll-int-win”.

Example: “ll-int-osx”.

Example: “ll-win-timeline-based”.

Example: “ll-osx-timeline-based”.

Example: “ll-int-win-doc”.

Example: “ll-int-apk”.

Example: “ll-web”.

Example: “ll-doc”.

Format of the analysis report. This value can be used to determine the expected values in addition to **uuid** and **format**. For details on each report format, see *Report Format ll-int-win*, *Report Format ll-int-osx*, *Report Format ll-int-win-doc*, *Report Format ll-int-apk*, *Report Format ll-int-archive*, *Report Format ll-web*, *Report Format ll-static*, *Report Format ll-ioc-json*, *Report Format ll-win-timeline-based*, *Report Format ll-osx-timeline-based*, *Report Format ll-pcap*, and *Report Format ll-flash*, *Report Format ll-doc*.

Note: Reports in format *ll-win-timeline-thread-based* are identical to reports in format *ll-win-timeline-based*. The latter is merely a backwards-compatible naming for the former.

- **major_version.** *Type:* Integer.

Example: 1.

Major part of the report version with format *<major>.<minor>.<build>*.

- **minor_version.** *Type:* Integer.

Example: 1.

Minor part of the report version with format *<major>.<minor>.<build>*.

- **build_version.** *Type:* Integer.

Example: 0.

Build part of the report version with format *<major>.<minor>.<build>*.

3.2 Report Format

Some components of an analysis report apply to multiple report formats. Each individual report format can extend this information with additional data, but follows the basic concepts described below.

Examples of such shared components is are the representation of *analysis subjects* or *file metadata*.

3.2.1 Analysis Subject Format

The analysis engine will monitor all analysis subjects, such as the originally started program and all child processes or processes that a monitored program interacts with, and then list any security relevant data.

Analysis subject contents

- **overview.** *Type:* Dictionary.

Overview of the analysis subject.

- **id.** *Type:* Integer.

Example: 2.

Identifier of the analysis subject within the analysis report.

- **parent_id: (optional).** *Type:* Integer.

Example: 2.

Identifier that indicates which analysis subject is responsible for the execution of the current analysis subject; If not present, it indicates that the monitoring of the current analysis subject did not depend on any other analysis subject (for example, the originally started program does not include this field). If present, it identifies an analysis subject within the report. For example, if *parent_id* is equal to 2, it means that the current analysis subject was monitored because it was started or interacted with the analysis subject whose identifier (as specified by the *id* field) has value 2.

- **ext_info: (optional).** *Type:* File-Info; see *Static File Information*.

Static information on the analyzed file.

3.2.2 Static File Information

The analysis engine extracts static file information for most files manipulated during the analysis run and associates this information with all files in the report.

File information contents

- **md5: (optional).** *Type:* Hexadecimal string.
Example: “748cb82987899a164c2f6e7985fffec5”.
A md5 hash of a file content.
- **sha1: (optional).** *Type:* Hexadecimal string.
Example: “066e791be6fb28063fc643cea658bf70d193b895”.
A sha1 hash of a file content.
- **file_info: (optional).** *Type:* String.
Example: “MS Windows shortcut”.
A text description of file type.
- **size: (optional).** *Type:* Integer.
Example: 1233.
A file size in bytes.

3.2.3 Analysis Metadata Format

During the analysis run, the analysis engine extracts the metadata that is available for download.

Metadata contents

- **metadata_type.** *Type:* String.
Example: “screenshot”.
Example: “traffic_capture”.
Example: “generated_file”.
Example: “memory_dump”.
Example: “process_dump”.
Example: “analysis_subject”.
Example: “extracted_file”.
Type of metadata.
- **delete_date: (optional).** *Type:* Date-Time.
Example: “2013-10-05 14:22:02”.
Analysis metadata is deleted according to data-retention policies. If a metadata file becomes unavailable, this field contains the delete date.
- **analysis_subject_id: (optional).** *Type:* Integer.
Example: 2.
Index of the analysis subject that metadata is associated with. Applies to metadata_types “memory_dump”, “analysis_subject”, and “process_dump”.

- **yara_signature_hits: (optional).** *Type:* List of strings.

Example: “SignatureName1”, “SignatureName2”

Yara signatures that matched on the analysis metadata. Applies to metadata_types “memory_dump”, “generated_file”, “analysis_subject”, and “process_dump”.

- **description: (optional).** *Type:* String.

Example: “Memory snapshot of 32-bit process”

Short description of the analysis metadata. Applies to metadata_types “memory_dump”, “generated_file”, “analysis_subject”, and “process_dump”.

- **embedded_shellcode: (optional).** *Type:* Boolean.

True if the buffer contains shellcode. Applies to metadata_type “memory_dump”.

- **ext_info: (optional).** *Type:* File-Info; see *Static File Information*.

Static file information. Applies to metadata_type “analysis_subject”.

- **timestamp: (optional).** *Type:* Integer.

Example: 50.

Number of seconds after start of analysis run at which the screenshot was taken. Applies to metadata_type “screenshot”.

3.2.4 Network Traffic Format

During the analysis run, the analysis engine extracts information about network connections observed during the analysis. The report will distinguish between different protocols and extract protocol-specific information.

Examples for supported protocols are *TCP*, *UDP*, *HTTP*, or *SMTP*.

Network connection contents

Network connections using a protocol that is not parsed into a more specific protocol type.

- **protocol: (optional).** *Type:* String.

Example: “TCP”.

Name of highest-level protocol recognized.

- **src_ip.** *Type:* IP address.

Example: “192.168.0.2”.

Source IP address of the connection.

- **src_port: (optional).** *Type:* Integer.

Example: 1036.

Source port of the connection, applies only to protocols using ports (such as “TCP” or “UDP”).

- **dst_ip.** *Type:* IP address.

Example: “2.2.2.2”.

Destination IP address of the connection.

- **dst_port: (optional).** *Type:* Integer.
Example: 80.
Destination port of the connection, applies only to protocols using ports (such as “TCP” or “UDP”).
- **type.** *Type:* String.
Example: “outgoing”.
Direction of the connection establishment. “incoming” or “listening”.
- **conversation: (optional).** *Type:* List of communication tuples; see below.
Example: (“message1”, “answer1”), (“”, “answer2”).
List of tuples containing incoming and outgoing message data.

HTTP connection contents

Network connections identified to use the HTTP protocol. This type extends the *network connection* type.

- **url.** *Type:* String.
Example: “GET /search?q=0 HTTP/1.0”.
Request line sent to server.
- **response_headers: (optional).** *Type:* Key-value storage.
Example: “type”=“HTTP Response”.
Each element corresponds to an HTTP header (name and value) sent to the server.
- **response_protocol_version: (optional).** *Type:* String.
Example: “1.1”.
Protocol version used in the server’s response.
- **response_status: (optional).** *Type:* String.
Example: “200”.
HTTP status code returned by the server.

IRC connection contents

Network connections identified to use the IRC protocol. This type extends the *network connection* type.

- **irc_channel: (optional).** *Type:* String.
Example: “channel1”.
IRC channel used in the communication.
- **channel_password: (optional).** *Type:* String.
Example: “channel_password1”.
Channel-password used for authenticating to the IRC channel.
- **irc_user: (optional).** *Type:* String.
Example: “user1”.
User used for authenticating the communication.

- **irc_password: (optional).** *Type:* String.
Example: “password1”.
Password used for authenticating the communication.
- **irc_nick: (optional).** *Type:* String.
Example: “nick1”.
IRC nick(name) used in the communication.

FTP connection contents

Network connections identified to use the FTP protocol. This type extends the *network connection* type.

- **ftp_login: (optional).** *Type:* String.
Example: “credentials1”.
FTP credentials used in the communication.

DNS query contents

- **hostname.** *Type:* String.
Example: “www.example.com”.
Name of host in query.
- **results: (optional).** *Type:* List of IP addresses.
Example: “1.1.1.1”, “2.2.2.2”.
IP address returned in result (if any).
- **response_flags: (optional).** *Type:* List of strings.
Example: “nxdomain”.
DNS flags set in the response.

SMTP connection contents

Network connections identified to use the SMTP protocol. This type extends the *network connection* type.

- **email_subject: (optional).** *Type:* String.
Example: “Email Subject”.
Email subject of a sent message in the SMTP conversation.
- **sender: (optional).** *Type:* String.
Example: “john.doe@example.com”.
Sender email address of a sent message in the SMTP conversation.
- **recipients: (optional).** *Type:* String.
Example: “Jane <jane.doe@example.com>”.
Sender recipient addresses of a sent message in the SMTP conversation.

Address scan contents

Network address scans.

- **subnet: (optional).** *Type:* String.
Example: “192.168.0.0/16”.
Network targeted by the network address scan.
- **remote_port: (optional).** *Type:* Integer.
Example: 80.
Destination port used in the network address scan.

3.3 Report Format *ll-int-win*

This analysis report format applies to a dynamic analysis run on a Microsoft Windows platform.

In addition to the report fields that all report formats share (see *Analysis Report Format*), the report contains a number of different fields with details about the analysis run.

Report contents

- **remarks: (optional).** *Type:* Dictionary.
 - **info (optional).** *Type:* List of strings.
Example: “Text1”,”Text2”.
A list of information strings concerning the analysis run.
 - **warning (optional).** *Type:* List of strings.
Example: “Text1”,”Text2”.
A list of warning strings concerning the analysis run.
- **overview.** *Type:* Dictionary.
 - **analysis_engine.** *Type:* String.
Example: “LLama - WindowsXP”.
Name of the analysis engine used for generating the result.
 - **analysis_engine_version.** *Type:* String.
Example: “1.2.4”.
Version of the analysis engine used for generating the result.
 - **analysis_start.** *Type:* Date-Time.
Example: “2013-10-05 14:21:01.928894”.
Start timestamp of the analysis run.
 - **analysis_end.** *Type:* Date-Time.
Example: “2013-10-05 14:22:02.935794”.
End timestamp of the analysis run.
- **analysis_subjects.** *Type:* List of analysis subjects; see *Windows Analysis Subject Format*.
A list of programs monitored during the analysis run.

- **analysis_metadata: (optional).** *Type:* List of analysis metadata; see *Windows Analysis Metadata Format*.
A list of artifacts generated during the analysis run. See `get_result_artifact()` for retrieving this metadata.
- **randomized_registry_values: (optional).** *Type:* List of registry keys; see *registry_reads*.
A list of Microsoft Windows Registry values that the analysis engine randomized during the analysis run to avoid detection by the analysis subject. For the format of each value, refer to *registry_reads*.
- **url_summary: (optional).** *Type:* List of strings.
Example: “http://www.example1.com”, “http://www.example2.com”.
Network summary of contacted URLs during analysis run.

3.3.1 Windows Analysis Subject Format

The analysis engine will monitor all analysis subjects, such as the originally started program and all child processes or processes that a monitored program interacts with, and then list any security relevant data.

This type extends the *Analysis Subject Format* type with additional information on Windows analysis subjects.

Analysis subject contents

- **overview.** *Type:* Dictionary.
Overview of the analysis subject. In addition to the base format contents, the following elements are extracted:
 - **process.** *Type:* Process; see *Windows Process*.
Information on the Windows process.
 - **ext_info: (optional).** *Type:* File-Info; see *Static File Information*.
Static information on the process image.
- **console_output: (optional).** *Type:* Dictionary.
Console output of the program.
 - **stdout: (optional).** *Type:* String.
Example: “text written to stdout”.
Program output written to default console.
 - **stderr: (optional).** *Type:* String.
Example: “text written to error console”.
Program output written to error console.
- **opened_windows: (optional).** *Type:* List of GUI-windows; see below.
A list of GUI windows opened by the analysis subject.
 - **title.** *Type:* String.
Example: “Documents and Settings”.
Window title content.

- **text:** *Type:* String.
Example: “FolderView”.
Window text content.
- **loaded_libraries: (optional).** *Type:* List of libraries; see below.
List of library files loaded by the analysis subject.
 - **filename:** *Type:* String.
Example: “C:\windows\syswow64\ole32.dll”.
Path to the library loaded by the analysis subject.
- **registry_reads: (optional).** *Type:* List of registry keys; see below.
A list of registry keys read by the analysis subject.
 - **key.** *Type:* String.
Example: “HKLM\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\DRIVERS32”.
A registry key.
 - **value: (optional).** *Type:* String.
Example: “wave9”.
A registry value.
 - **data: (optional).** *Type:* String or Integer.
Example: 1, “mso.dll”.
A data of registry value.
- **registry_writes: (optional).** *Type:* List of registry keys; see [registry_reads](#).
A list of registry keys written by the analysis subject.
- **registry_deletions: (optional).** *Type:* List of registry keys; see [registry_reads](#).
A list of registry keys deleted by the analysis subject.
- **file_reads: (optional).** *Type:* List of files; see below.
A list of files read by the analysis subject
 - **filename.** *Type:* String.
Example: “desktop.ini”.
A file name. Could be absolute or relative path.
 - **abs_path: (optional).** *Type:* String.
Example: “C:\Users\desktop.ini”.
An absolute path the file.
 - **ext_info: (optional).** *Type:* File-Info; see [Static File Information](#).
Static file information.
- **file_writes: (optional).** *Type:* List of files; see [file_reads](#).
A list of files written by the analysis subject.

- **file_deletes: (optional).** *Type:* List of files; see *file_reads*.
A list of files deleted by the analysis subject.
- **file_hardlinks: (optional).** *Type:* List of files; see *file_reads*.
A list of files hardlinked by the analysis subject.
- **file_renames: (optional).** *Type:* List of files; see *file_reads*.
A list of files renamed by the analysis subject.
- **file_queries: (optional).** *Type:* List of files; see *file_reads*.
A dictionary of files which status was being queried by the analysis subject. Key - file object.
Value - status (int value)
- **file_searches: (optional).** *Type:* List of strings.
A list of files searched for by the analysis subject.
- **process_interactions: (optional).** *Type:* List of process-interactions; see below.
A list of processes the analysis subject interacts with.
In addition to the fields of type *Windows Process*, each element contains the operation(s) performed:
 - **operations:** *Type:* List of strings.
Example: “create_thread”, “write_mem”.
The type of operations performed on the remote process. Possible values are:
 - * “create_process”: Create a process.
 - * “terminate_process”: Terminate a process.
 - * “create_thread”: Create a thread.
 - * “terminate_thread”: Terminate a thread.
 - * “read_mem”: Read from the process memory.
 - * “write_mem”: Write to the process memory.
- **mutex_creates: (optional).** *Type:* List of mutexes; see below.
A list of mutex synchronization objects created by the analysis subject.
 - **mutex_name:** *Type:* String.
Example: “Mutex1”.
Name of the mutex synchronization object.
- **mutex_opens: (optional).** *Type:* List of mutexes; see *mutex_creates*.
A list of mutex synchronization objects opened by the analysis subject.
- **raised_exceptions: (optional).** *Type:* List of exceptions; see below.
A list of exceptions raised by the analysis subject.
 - **addr: (optional).** *Type:* Hexadecimal number.
Example: 0x7c832297.
Instruction address raising the exception.

- **code: (optional).** *Type:* Hexadecimal number.
Example: 0xc0000005.
Microsoft Windows exception code.
- **name: (optional).** *Type:* String.
Example: “STATUS_ACCESS_VIOLATION”.
Microsoft Windows exception name.
- **exception_name: (optional).** *Type:* String.
Example: “Exception 0xc0000005 (STATUS_ACCESS_VIOLATION) at 0x7c832297 (subject_id: 2)”.
Full exception information.
- **exception_count: (optional).** *Type:* Integer.
Example: 1.
Number of exception occurrences.
- **idt_hooks: (optional).** *Type:* List of the modified IDT entries; see below.
A list of the modified by the analysis subject IDT entries.
 - **vector: (optional).** *Type:* Integer number.
Example: 113.
An interruption number.
 - **isr_address: (optional).** *Type:* Hexadecimal number.
Example: 0xfffff800019fc710.
An original virtual address of the ISR (Interrupt Service Routines).
 - **new_isr_address: (optional).** *Type:* Hexadecimal number.
Example: 0xfffff880041f1b9d.
A new virtual address of the ISR (Interrupt Service Routines).
 - **isr_module_name: (optional).** *Type:* String.
Example: “c:\windows\system32\ntkrnlmp.exe”.
A module which contains the original ISR (Interrupt Service Routines).
 - **new_isr_module_name: (optional).** *Type:* String.
Example: “c:\windows\system32\hellodrv.sys”.
A module which contains a new ISR (Interrupt Service Routines).
 - **modification_sources: (optional).** *Type:* List of the sources of the IDT modifications; see *modification_sources*.
- **ssdt_hooks: (optional).** *Type:* List of the modified SSDT or Shadow SSDT entries; see below.
A list of the modified by the analysis subject SSDT/Shadow SSDT entries.
 - **service_index: (optional).** *Type:* Hexadecimal number.
Example: 0x30.
A service index in the service table.

- **service_address: (optional).** *Type:* Hexadecimal number.
Example: 0x8057fc60.
An original address of the service function.
- **new_service_address: (optional).** *Type:* Hexadecimal number.
Example: 0xf7a574b1.
A new address of the service function.
- **service_module_name: (optional).** *Type:* String.
Example: “c:\windows\system32\ntkrnlmp.exe”.
An original module which contains the service function.
- **new_service_module_name: (optional).** *Type:* String.
Example: “c:\windows\system32\ntkrnlmp.exe”.
A new module which contains the service function.
- **service_function_name: (optional).** *Type:* String.
Example: “NtCreateProcessEx”.
A name of the modified service function.
- **table_name: (optional).** *Type:* String.
Example: “system service table”.
A name of the system table.
- **modification_sources: (optional).** *Type:* List of the sources of the SSDT modification; see *modification_sources*.
- **kernel_memory_hooks: (optional).** *Type:* List of the modified kernel memory areas; see below.
A list of the modified critical system areas.
 - **kernel_address: (optional).** *Type:* Hexadecimal number.
Example: 0x8056cdc0.
The modified kernel address.
 - **kernel_module_name: (optional).** *Type:* String.
Example: “c:\windows\system32\ntkrnlmp.exe”.
The modified kernel module name.
 - **area_name: (optional).** *Type:* String.
Example: “kernel image section PAGE”.
A name of the modified critical area.
 - **kernel_function_name: (optional).** *Type:* String.
Example: “NtCreateFile”.
A name of the modified system function.
 - **modification_sources: (optional).** *Type:* List of the sources of the kernel memory modification; see *modification_sources*.
- **modification_sources: (optional).** *Type:* List of the sources of the code or memory modification.

- **virtual_address: (optional).** *Type:* Hexadecimal number.
Example: 0xffff880041f1b9d.
The virtual address of an instruction which overwrote an original code or memory.
- **modifier_module_name: (optional).** *Type:* String.
Example: “c:\windows\system32\hellodrv.sys”.
The module containing the instruction which overwrote an original code or memory.
- **service_creates: (optional).** *Type:* List of services. See below.
A list of the services created by the analysis subject.
 - **service_file: (optional).** *Type:* File. see [file_reads](#).
 - **service_name: (optional).** *Type:* String.
A display name to be used by user interface programs to identify the service.
 - **start_type: (optional).** *Type:* Integer.
Example: 0x00000002 (SERVICE_FILE_SYSTEM_DRIVER)
The service type.
 - **start_type: (optional).** *Type:* Integer.
Example: 0x00000000 (SERVICE_BOOT_START)
The service start options.
- **service_starts: (optional).** *Type:* List of services; see [service_creates](#).
A list of services started by the analysis subject.
- **service_stops: (optional).** *Type:* List of services; see [service_creates](#).
A list of services stopped by the analysis subject.
- **service_changes: (optional).** *Type:* List of services changes; see below.
A list of the services changed by the analysis subject.
 - **service_name: (optional).** *Type:* String.
Example: “WNKiserv”
A display name to be used by user interface programs to identify the service.
 - **change: (optional).** *Type:* String.
A change of the service made by the analysis subject.
- **driver_loads: (optional).** *Type:* Services; see [service_creates](#).
A list of the drives loaded by the analysis subject.
- **driver_unloads: (optional).** *Type:* Services; see [service_creates](#).
A list of the drives unloaded by the analysis subject.
- **dns_queries: (optional).** *Type:* List of DNS queries; see [DNS query](#).
List of DNS queries done by the analysis subject.

- **network_connections: (optional).** *Type:* List of network connections; see *network connection*.
List of network connections done by the analysis subject using a protocol that is not parsed into a more specific protocol type.
- **http_conversations: (optional).** *Type:* List of HTTP connections; see *HTTP connection*.
List of network connections identified to use the HTTP protocol done by the analysis subject.
- **irc_conversations: (optional).** *Type:* List of IRC connections; see *IRC connection*.
List of network connections identified to use the IRC protocol done by the analysis subject.
- **ftp_conversations: (optional).** *Type:* List of FTP connections; see *FTP connection*.
List of network connections identified to use the FTP protocol done by the analysis subject.
- **smtp_conversations: (optional).** *Type:* List of SMTP connections; see *SMTP connection*.
List of network connections identified to use the SMTP protocol done by the analysis subject.
- **address_scans: (optional).** *Type:* List of network address scans; see *address scan*.
List of network address scans done by the analysis subject.
- **downloaded_files: (optional).** *Type:* List of file-download tuples; see below.
List of files that were downloaded using the Microsoft Windows file-download API functions. Each element is a tuple of file-origin URL and a File element (see *file_reads*).
Note: This list does not contain files downloaded using other mechanisms or protocol (such as HTTP). Those are listed in the corresponding network section.
- **pe_images: (optional).** *Type:* List of PE images; see below.
A list of PE images found in the memory of the analysis subject.
 - **image.** *Type:* PE image; see *Portable Executable Image*.
Process image information extracted when included in the analysis.
 - **image_diff: (optional).** *Type:* PE image; see *Portable Executable Image*.
Process image information extracted at program termination or analysis end.
- **memory_blocks: (optional).** *Type:* List of memory-blocks; see below.
A list of allocated memory regions found in the memory of the analysis subject.
 - **name: (optional).** *Type:* String.
Example: “mem_b67f3190f04083ac1e0189307f4d64d4”.
A name of the memory block. Format: mem_<md5>.
 - **size: (optional).** *Type:* Integer.
A size of the memory block in bytes.
 - **start_va: (optional).** *Type:* Integer.
A VA to where the memory block starts.
 - **end_va: (optional).** *Type:* Integer.
A VA to where the memory block ends.

- **access: (optional).** *Type:* Hexadecimal string.
A set of flags that indicate the memory block’s attributes (such as code/data, readable, or writable).
- **number_of_executed_pages: (optional).** *Type:* Integer.
A number of executed pages in the memory block.
- **executed_pages: (optional).** *Type:* List of integers.
A list of VA of executed pages.
- **dist_bytes_vector: (optional).** *Type:* Hexadecimal string.
An internal field used by an analysis engine.
- **average_bytes: (optional).** *Type:* Hexadecimal string.
An internal field used by an analysis engine.
- **autocorrelation: (optional).** *Type:* Floating-point number.
An autocorrelation of the memory block data.
- **entropy: (optional).** *Type:* Floating-point number.
An entropy of the memory block data.
- **embedded_pe_header.** *Type:* Boolean.
True if we recognized a PE image header in the memory block.
- **number_of_strings: (optional).** *Type:* Integer.
A number of found strings in the memory block.
- **strings: (optional).** *Type:* List of strings.
Example: “SSP3FR.DLL”,”ROOF”,”Help Button”.
A list of strings found in the memory block.
- **md5.** *Type:* Hexadecimal string.
A md5 hash of a content.
- **strings_lists: (optional).** *Type:* List of named strings-lists; see below.
A list of named strings-lists. The name identifies a type of the strings in the list.
 - **name: (optional).** *Type:* String.
Example: “heap_strings”.
A name of the string list.
 - **strings: (optional).** *Type:* List of strings.
Example: “ProgramData=C:\ProgramData”,”NUMBER_OF_PROCESSORS=1”,”ncacn_ip_tcp”.
A list of strings.
- **patched_sleeps: (optional).** *Type:* List of patched sleep values; see below.
A list of patched sleep values. It is an anti-evasion technique which changes a waiting period (if it is too long) for a sleep function and timers.

- **count: (optional).** *Type:* Integer.

Example: 1.

A number of times the sleep function was called.

- **new_value: (optional).** *Type:* Integer.

Example: 5.

A new value of the waiting period in seconds.

- **old_value: (optional).** *Type:* Integer.

Example: 3600.

A old value of the waiting period in seconds.

- **strings_comparisons: (optional).** *Type:* List of string comparisons; see below.

A list of string comparisons.

- **name: (optional).** *Type:* String.

Example: “shlwapi.dll.StrStr_generic”.

A possible name of the string comparison function. Usually defined by a flirt signature *flirt_signatures*.

- **src_string: (optional).** *Type:* String.

Example: “C:\Users\Public\Desktop”.

A comparable string 1.

- **dst_string: (optional).** *Type:* String.

Example: “%SYSTEMROOT%”.

A comparable string 2.

- **src_sources: (optional).** *Type:* List of strings.

Example: “\Registry\Machine\Software\Classes.dot\Icon”, “Command line”.

A a list of possible sources (from where this string could be read) for string 1.

- **dst_sources: (optional).** *Type:* List of strings.

Example: “\Registry\Machine\Software\Classes.dot\Icon”, “Command line”.

A a list of possible sources (from where this string could be read) for string 2.

- **frequent_api_calls: (optional).** *Type:* List of the frequent API calls; see below.

A list of the frequent API calls.

- **name: (optional).** *Type:* String.

Example: “NtOpenThreadToken”.

A name of the API function which was frequently called.

- **count: (optional).** *Type:* String.

Example: 31440.

A number of times the API function was called.

- **pid: (optional).** *Type:* Integer.
Example: 145.
Windows process identifier (TID) of the calling process.
- **tid: (optional).** *Type:* Integer.
Example: 167.
Windows thread identifier (TID) of the calling thread.
- **yara_signatures: (optional).** *Type:* List of the Yara signatures; see below.
A list of the Yara signatures which hit on the analysis subject.
 - **name: (optional).** *Type:* String.
Example: “apt_win_generic_imageStego”.
A name of the Yara signatures.
 - **score: (optional).** *Type:* Integer.
Example: 75.
A score which defines how dangerous the analysis subject according to the Yara signature. Possible range from 0 (benign) to 100 (malicious).
 - **internal: (optional).** *Type:* Boolean.
If true the signature is only for an internal usage.
- **flirt_signatures: (optional).** *Type:* List of the Yara signatures; see below.
A list of the flirt signatures which hit on the analysis subject. The flirt signatures recognize a known functions.
 - **name: (optional).** *Type:* String.
Example: “__ascii_strnicmp”
Name of the signature.
- **keyboard_capture: (optional).** *Type:* List of keyboard actions captured; see below.
A list of keyboard actions sent during the analysis which were seen on malware behavior.
 - **word: (optional).** *Type:* String.
Example: “8989-8408-5161-4765”
Word captured by sample during analysis.
 - **word_type: (optional).** *Type:* String.
Example: “Credit Card”
Type of word that was captured.

3.3.2 Windows Process

Information on a Windows process.

Windows process contents

- **process_id.** *Type:* String.
Example: “1376”.
Windows process identifier (PID).
- **executable: (optional).** *Type:* File; see *file_reads*.
Process image information.
- **arguments: (optional).** *Type:* String.
Example: “C:\subject.exe arg1 arg2”.
Full command line used to start the analysis subject.
- **bitsize: (optional).** *Type:* Integer.
Example: 32.
Process bit-size (32bit or 64bit process).
- **analysis_subject_id: (optional).** *Type:* Integer.
Example: 2.
Identifier of the analysis subject within the analysis report if the process belongs to an analysis subject monitored in the analysis run.

3.3.3 Portable Executable Image

A process image extracted during the analysis run.

Process image contents

- **name: (optional).** *Type:* String.
Example: “pe_i_5177683cb94a34bfb52142cfb1c49806”.
A name of the section.
Format:
 - “main” for the main PE image in the process.
 - “pe_i_<md5>” for the unknown PE image in the process recognized as loaded image.
 - “pe_f_<md5>” for the unknown PE image in the process recognized as non-loaded image.
- **image_base.** *Type:* Integer.
When the linker creates an executable, it assumes that the file will be memory-mapped to a specific location in memory.
- **directories_mask.** *Type:* Binary string.
A bit mask which shows a presence of each entry in DataDirectory of the image.
- **number_of_sections.** *Type:* Integer.
A number of sections in the image.
- **average_bytes.** *Type:* Hexadecimal string.
An internal field used by an analysis engine.
- **dll.** *Type:* Boolean.
true if the image is a Dynamic Load Library

- **entry_point.** *Type:* Integer.
An address where the loader will begin execution.
- **image_size.** *Type:* Integer.
A size of the region starting at the image base up to the end of the last section.
- **loaded.** *Type:* Boolean.
True if we recognized an image as a loaded PE image. False if we recognize as image as a file. (non loaded PE image).
- **md5.** *Type:* Hexadecimal string.
A md5 hash of a content.
- **sections: (optional).** *Type:* List of sections; see below.
PE image sections.
 - **md5_phys: (optional).** *Type:* Hexadecimal string.
A md5 hash of physical data of the section (offset : offset + physical_size).
 - **md5_virt: (optional).** *Type:* Hexadecimal string.
A md5 hash of virtual data of the section (virtual_address : virtual_address + virtual_size).
 - **name: (optional).** *Type:* String.
Example: “.text”.
A name of the section.
 - **physical_size: (optional).** *Type:* Integer.
A size of the section after it’s been rounded up to the file alignment size.
 - **offset: (optional).** *Type:* Integer.
A file-based offset of where the raw data emitted by the compiler or assembler can be found.
 - **virtual_size: (optional).** *Type:* Integer.
A size of the section after it’s been rounded up to the page alignment size.
 - **virtual_address: (optional).** *Type:* Integer.
A RVA to where the loader should map the section.
 - **characteristics: (optional).** *Type:* Hexadecimal string.
A set of flags that indicate the section’s attributes (such as code/data, readable, or writable).
 - **dist_bytes_vector: (optional).** *Type:* Hexadecimal string.
An internal field used by an analysis engine.
 - **average_bytes: (optional).** *Type:* Hexadecimal string.
An internal field used by an analysis engine.
 - **autocorrelation: (optional).** *Type:* Floating-point number.
An autocorrelation of the section data.
 - **entropy: (optional).** *Type:* Floating-point number.
An entropy of the section data.

- **number_of_strings: (optional).** *Type:* Integer.
A number of found strings in the section.
- **strings: (optional).** *Type:* List of strings.
Example: “SSP3FR.DLL”,”ROOF”,”Help Button”.
A list of strings found in the section.

3.3.4 Windows Analysis Metadata Format

During the analysis run, the analysis engine extracts the metadata that is available for download. This type extends the *Analysis Metadata Format* type.

Metadata contents

- **file: (optional).** *Type:* File; see *file_reads*.
Name of the file generated during the analysis run. Applies to metadata_type “generated_file”.

3.3.5 Windows Analysis Process Dumps (PE Snapshots)

The analysis engine extracts process dumps/snapshots of the programs under analysis whenever an interesting action is triggered. These snapshots are exported via analysis metadata; see *Windows Analysis Metadata Format*.

These snapshots are available in two formats:

- **process_snapshot:** This snapshot type is generated for all 32-bit or 64-bit processes that are tracked during the analysis. For each process, the process snapshot file contains allocated memory sections/areas mapped during the analysis, as well as metadata describing the placement of these different memory sections within the process memory and how the content/placement changes over time.
- **process_dump: (deprecated).** For 32-bit processes, the system exports PE files which can be analyzed with a wide variety of tools. Since the PE format is limited to describing code/data sections loaded at 32-bit addresses, this snapshot type is not applicable to 64-bit processes or programs that load data beyond the 32-bit address range.

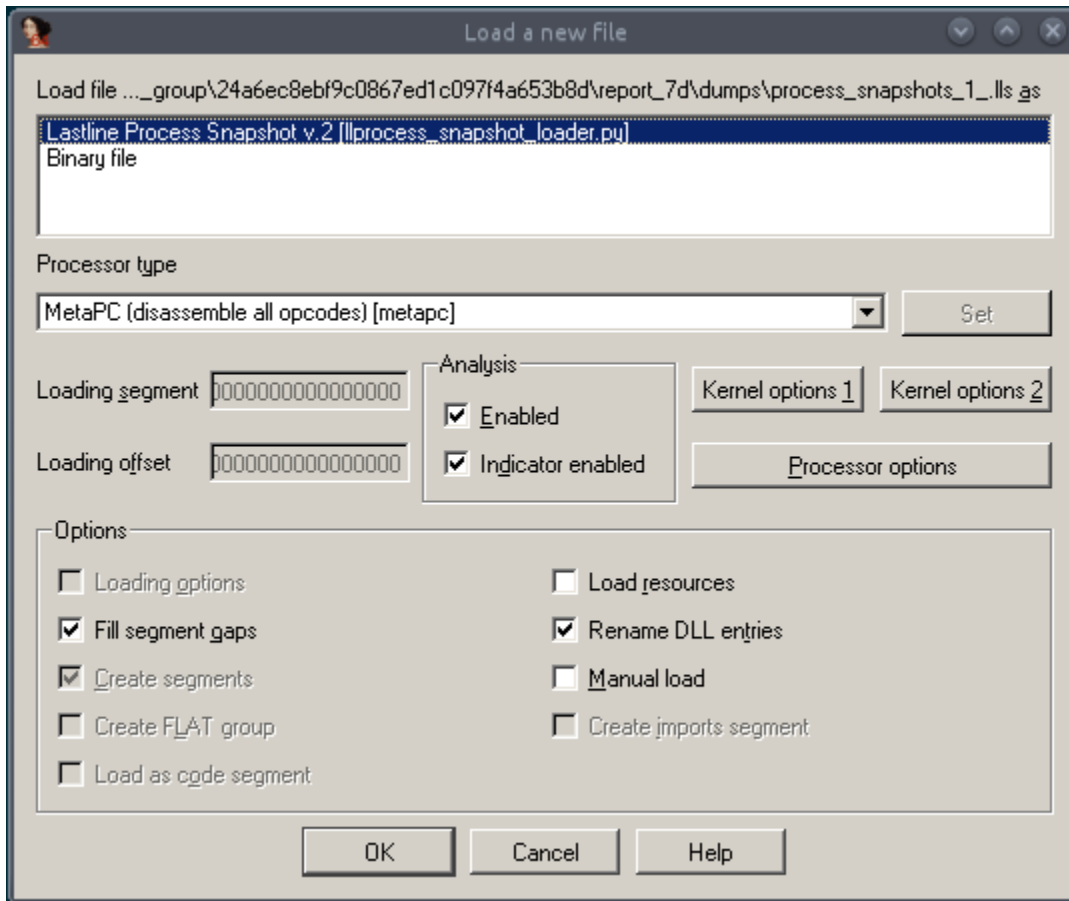
At the same time, this snapshot type does not support multiple snapshot states within one file, as supported by files in the *process_snapshot* format.

3.3.6 IDA Pro Integration

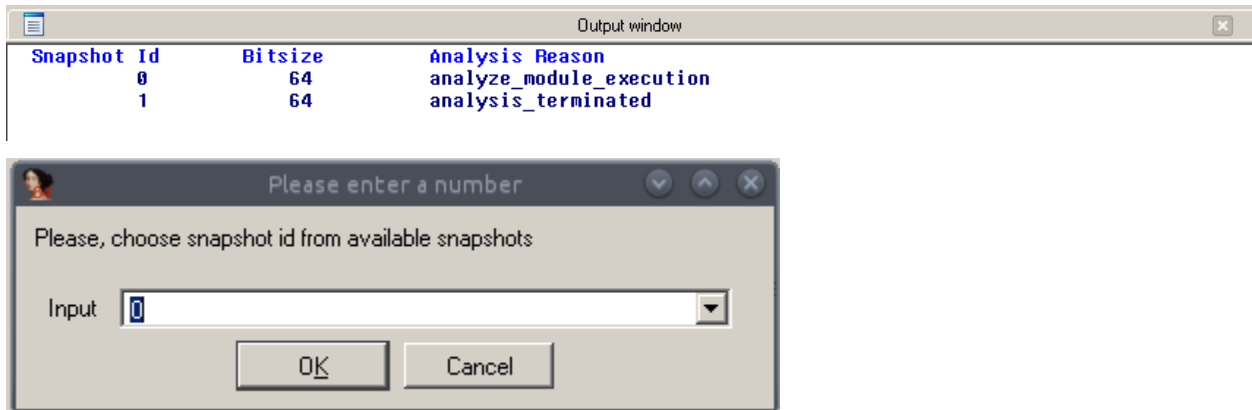
When loading a *process_snapshot* file in IDA Pro, the code is annotated with runtime information extracted during the dynamic analysis, such as API function exports, program entry-points, or instruction addresses of interesting code regions.

To get started, download the Python loader available at https://analysis.lastline.com/analysis/api-docs/examples/lprocess_snapshot_loader.py to the IDA Pro directory *<IDA dir>/loaders/*.

With the loader added to IDA Pro, the program offers a new loader window for process snapshots generated by the Lastline analysis system:



Select the *Lastline Process Snapshot* option. The IDA loader displays the snapshots available in the input file, as well as additional metadata for each individual snapshot; select the snapshot for analysis and confirm with OK.



IDA Pro assembles the data and code blocks as they were placed within the process memory at the point of the snapshot, and annotates the code with all available metadata: functions referenced from untrusted code are colored in blue, otherwise grey:

```

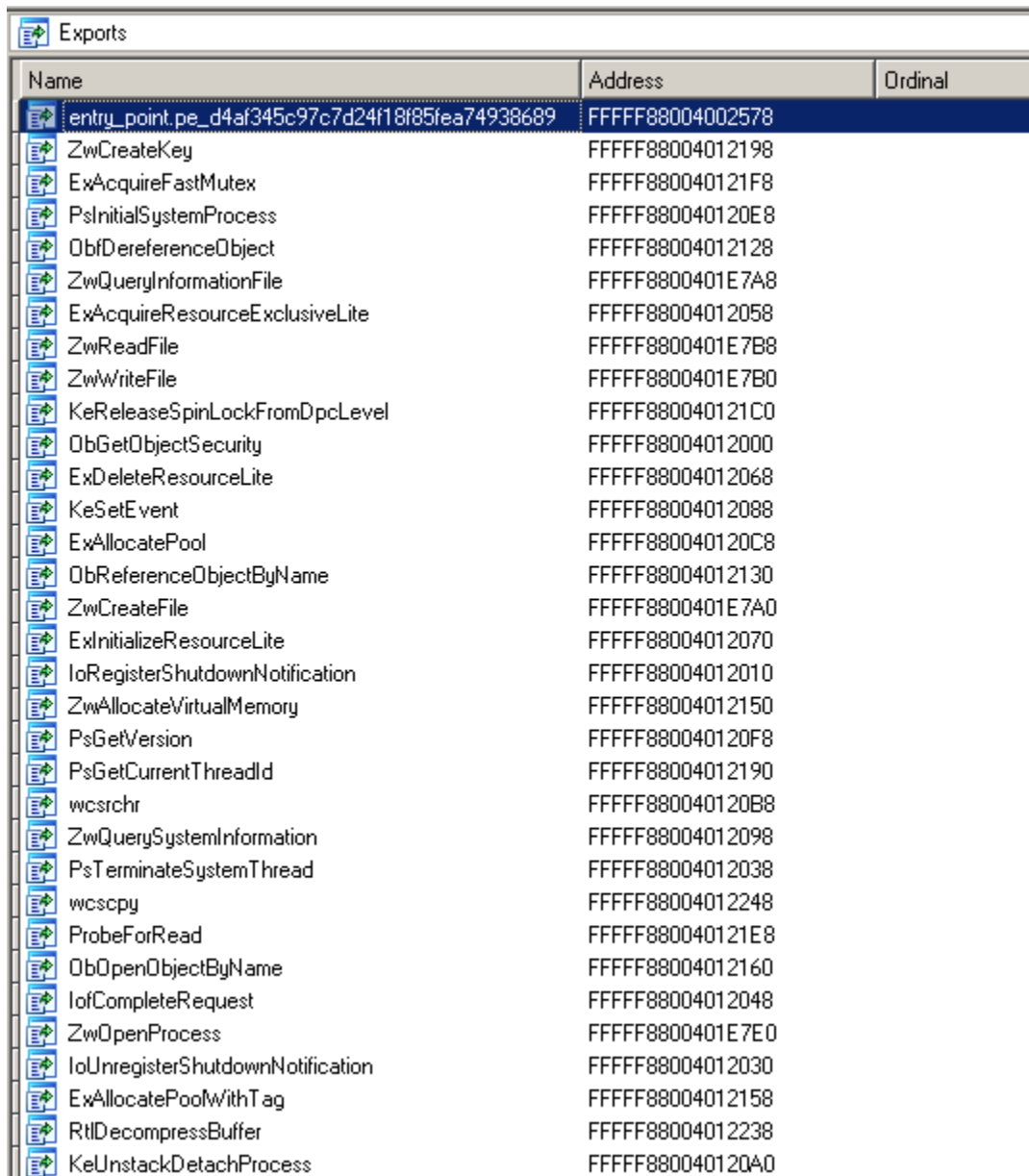
ntkrnlmp.exe:FFFFFF800018C7E00 ZwQueryInformationFile_ db ?
ntkrnlmp.exe:FFFFFF800018C7E00
ntkrnlmp.exe:FFFFFF800018C7E01 db 7 dup(?)
ntkrnlmp.exe:FFFFFF800018C7E08 dq 3 dup(?)
ntkrnlmp.exe:FFFFFF800018C7E20 ZwOpenKey db ?
ntkrnlmp.exe:FFFFFF800018C7E21 db 7 dup(?)
ntkrnlmp.exe:FFFFFF800018C7E28 dq 3 dup(?)
ntkrnlmp.exe:FFFFFF800018C7E40 ZwEnumerateValueKey db ?
ntkrnlmp.exe:FFFFFF800018C7E41 db 7 dup(?)
ntkrnlmp.exe:FFFFFF800018C7E48 dq 0Fh dup(?)
ntkrnlmp.exe:FFFFFF800018C7EC0 ZwQueryValueKey_ db ?
ntkrnlmp.exe:FFFFFF800018C7EC0
ntkrnlmp.exe:FFFFFF800018C7EC0
ntkrnlmp.exe:FFFFFF800018C7EC1 db 7 dup(?)
ntkrnlmp.exe:FFFFFF800018C7EC8 dq 3 dup(?)
ntkrnlmp.exe:FFFFFF800018C7EE0 ZwAllocateVirtualMemory_ db ?
ntkrnlmp.exe:FFFFFF800018C7EE1 db 7 dup(?)
ntkrnlmp.exe:FFFFFF800018C7EE8 dq 3 dup(?)
ntkrnlmp.exe:FFFFFF800018C7F00 ZwQueryInformationProcess_ db ?
ntkrnlmp.exe:FFFFFF800018C7F00
ntkrnlmp.exe:FFFFFF800018C7F01 db 7 dup(?)
ntkrnlmp.exe:FFFFFF800018C7F08 dq 0Bh dup(?)
ntkrnlmp.exe:FFFFFF800018C7F60 ZwSetInformationProcess db ?
ntkrnlmp.exe:FFFFFF800018C7F61 db 7 dup(?)
ntkrnlmp.exe:FFFFFF800018C7F68 dq 3 dup(?)
ntkrnlmp.exe:FFFFFF800018C7F80 ZwCreateKey_ db ?
ntkrnlmp.exe:FFFFFF800018C7F80
ntkrnlmp.exe:FFFFFF800018C7F81 db 7 dup(?)
ntkrnlmp.exe:FFFFFF800018C7F88 dq 0Fh dup(?)
ntkrnlmp.exe:FFFFFF800018C8000 ZwQueryInformationToken db ?
ntkrnlmp.exe:FFFFFF800018C8001 db 7 dup(?)
ntkrnlmp.exe:FFFFFF800018C8008 dq 13h dup(?)
ntkrnlmp.exe:FFFFFF800018C80A0 ZwOpenProcess_ db ?
ntkrnlmp.exe:FFFFFF800018C80A0
ntkrnlmp.exe:FFFFFF800018C80A1 db 7 dup(?)
ntkrnlmp.exe:FFFFFF800018C80A8 dq 3 dup(?)
ntkrnlmp.exe:FFFFFF800018C80C0 ZwSetInformationFile db ?
ntkrnlmp.exe:FFFFFF800018C80C1 db 7 dup(?)
ntkrnlmp.exe:FFFFFF800018C80C8 dq 3 dup(?)
ntkrnlmp.exe:FFFFFF800018C80E0 ZwMapViewOfSection db ?

```

Loaded system modules are represented as empty segments with exported functions as named references:

| Name | Start | End | R | W | X | D | L | Align | Base | Type | Class | AD | es | ss | ds | fs | gs |
|----------------------------|-------------------|------------------|---|---|---|---|---|-------|------|--------|-------|----|------|----------|------|----------|----------|
| kdcnm.dll | FFFFF800016D8000 | FFFFF800016D95FC | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| hal.dll | FFFFF80001806000 | FFFFF8000184A5D8 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| ntkrnlmp.exe | FFFFF8000184F000 | FFFFF80001DF65C0 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| atapi.sys | FFFFF88000C00000 | FFFFF88000C1C4A0 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| pshed.dll | FFFFF88000C8C000 | FFFFF88000C99244 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| clfs.sys | FFFFF88000CA0000 | FFFFF88000CED990 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| ci.dll | FFFFF88000CFE000 | FFFFF88000D08D10 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| pcidex.sys | FFFFF88000E64000 | FFFFF88000E6DFE8 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| wdfldr.sys | FFFFF88000F1B000 | FFFFF88000F25388 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| acpi.sys | FFFFF88000F2A000 | FFFFF88000F68FD8 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| wmilib.sys | FFFFF88000F81000 | FFFFF88000F86008 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| fltmgr.sys | FFFFF88001047000 | FFFFF88001083250 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| mstpc.sys | FFFFF88001047000 | FFFFF88001102040 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| cng.sys | FFFFF88001105000 | FFFFF88001140E30 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| tdi.sys | FFFFF88001177000 | FFFFF88001180758 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| ksecdd.sys | FFFFF88001200000 | FFFFF88001216590 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| netio.sys | FFFFF88001400000 | FFFFF8800145C110 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| videoprt.sys | FFFFF88001499000 | FFFFF880014B7163 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| watchdog.sys | FFFFF880014BE000 | FFFFF880014C9958 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| ndis.sys | FFFFF880014F9000 | FFFFF880015CF131 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| classpnp.sys | FFFFF88001616000 | FFFFF8800163D4D0 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| fwppkclnt.sys | FFFFF880018B4000 | FFFFF880018EC994 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| mup.sys | FFFFF880019AC000 | FFFFF880019A96C0 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| hwpolicy.sys | FFFFF880019AE000 | FFFFF880019B3008 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| usbport.sys | FFFFF88002200000 | FFFFF88002229060 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| ndistapi.sys | FFFFF8800229A000 | FFFFF8800229B9E8 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| termdd.sys | FFFFF8800276B000 | FFFFF880027744FC | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| rdss.sys | FFFFF8800277F000 | FFFFF880027C56E4 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| dxapi.sys | FFFFF88002A00000 | FFFFF88002A07374 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| ks.sys | FFFFF88002A3E000 | FFFFF88002A7D010 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| portcls.sys | FFFFF88002B8E000 | FFFFF88002B86008 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| dmk.sys | FFFFF88002B9B000 | FFFFF88002BAAD28 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| crashdmp.sys | FFFFF88002BC3000 | FFFFF88002BCC008 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| dump_dumpata.sys | FFFFF88002BD11000 | FFFFF88002BD4B60 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| srvnet.sys | FFFFF88003A00000 | FFFFF88003A29D20 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| mixsmb.sys | FFFFF88003B34000 | FFFFF88003B58B2C | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| text.section_817a85189... | FFFFF88004001000 | FFFFF88004011000 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| .msdata.section_b899562... | FFFFF88004011000 | FFFFF88004012000 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| .rdata.section_f7ab439a... | FFFFF88004012000 | FFFFF8800401E000 | ? | ? | ? | | L | byte | 0000 | public | DATA | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| .data.section_70529455... | FFFFF8800401E000 | FFFFF8800401F000 | ? | ? | ? | | L | byte | 0000 | public | DATA | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| .pdata.section_0fe5fe4a... | FFFFF8800401F000 | FFFFF88004020000 | ? | ? | ? | | L | byte | 0000 | public | DATA | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| text.section_81a3b4fa4... | FFFFF88004020000 | FFFFF8800402A000 | ? | ? | ? | | L | byte | 0000 | public | DATA | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| .edata.section_8b3c502... | FFFFF8800402A000 | FFFFF8800402B000 | ? | ? | ? | | L | byte | 0000 | public | DATA | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| .INIT.section_c9cdb7a4f... | FFFFF8800402B000 | FFFFF8800402C000 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| .rsrc.section_c9ac73a... | FFFFF8800402C000 | FFFFF8800402D000 | ? | ? | ? | | L | byte | 0000 | public | DATA | 32 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| .reloc.section_e408923... | FFFFF8800402D000 | FFFFF8800402E000 | ? | ? | ? | | L | byte | 0000 | public | DATA | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |
| spsys.sys | FFFFF88004152000 | FFFFF88004164400 | ? | ? | ? | | L | byte | 0000 | public | CODE | 64 | 0000 | FFFFF... | 0000 | FFFFF... | FFFFF... |

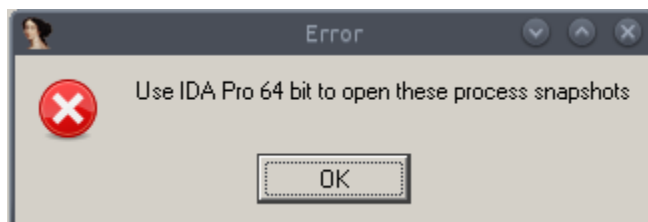
Unfortunately, IDA Pro currently does not provide the API to modify the program *import table*. As workaround, the loader exports a list of interesting entry points found by the analysis engine as well as references to API invocations from process memory as *export table*:



| Name | Address | Ordinal |
|---|-------------------|---------|
| entry_point.pe_d4af345c97c7d24f18f85fea74938689 | FFFFFF88004002578 | |
| ZwCreateKey | FFFFFF88004012198 | |
| ExAcquireFastMutex | FFFFFF880040121F8 | |
| PsInitialSystemProcess | FFFFFF880040120E8 | |
| ObfDereferenceObject | FFFFFF88004012128 | |
| ZwQueryInformationFile | FFFFFF8800401E7A8 | |
| ExAcquireResourceExclusiveLite | FFFFFF88004012058 | |
| ZwReadFile | FFFFFF8800401E7B8 | |
| ZwWriteFile | FFFFFF8800401E7B0 | |
| KeReleaseSpinLockFromDpcLevel | FFFFFF880040121C0 | |
| ObGetObjectSecurity | FFFFFF88004012000 | |
| ExDeleteResourceLite | FFFFFF88004012068 | |
| KeSetEvent | FFFFFF88004012088 | |
| ExAllocatePool | FFFFFF880040120C8 | |
| ObReferenceObjectByName | FFFFFF88004012130 | |
| ZwCreateFile | FFFFFF8800401E7A0 | |
| ExInitializeResourceLite | FFFFFF88004012070 | |
| IoRegisterShutdownNotification | FFFFFF88004012010 | |
| ZwAllocateVirtualMemory | FFFFFF88004012150 | |
| PsGetVersion | FFFFFF880040120F8 | |
| PsGetCurrentThreadId | FFFFFF88004012190 | |
| wcsrchr | FFFFFF880040120B8 | |
| ZwQuerySystemInformation | FFFFFF88004012098 | |
| PsTerminateSystemThread | FFFFFF88004012038 | |
| wcscpy | FFFFFF88004012248 | |
| ProbeForRead | FFFFFF880040121E8 | |
| ObOpenObjectByName | FFFFFF88004012160 | |
| IoCompleteRequest | FFFFFF88004012048 | |
| ZwOpenProcess | FFFFFF8800401E7E0 | |
| IoUnregisterShutdownNotification | FFFFFF88004012030 | |
| ExAllocatePoolWithTag | FFFFFF88004012158 | |
| RtlDecompressBuffer | FFFFFF88004012238 | |
| KeUnstackDetachProcess | FFFFFF880040120A0 | |

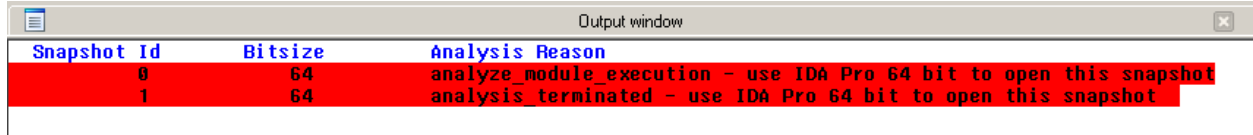
Unsupported IDA Pro Versions

When opening a process snapshot using an unsupported version of IDA Pro, the following error message is displayed:



This can happen, for example, when a 64-bit process snapshot is opened in an IDA Pro version limited to the analysis of 32-bit programs.

Likewise, the list of unavailable snapshots for this version of IDA Pro are marked accordingly:



| Snapshot Id | Bitsize | Analysis Reason |
|-------------|---------|---|
| 0 | 64 | analyze_module_execution - use IDA Pro 64 bit to open this snapshot |
| 1 | 64 | analysis_terminated - use IDA Pro 64 bit to open this snapshot |

3.4 Report Format *ll-int-osx*

This analysis report format applies to a dynamic analysis run on an OSX platform.

In addition to the report fields that all report formats share (see *Analysis Report Format*), the report contains a number of different fields with details about the analysis run.

Report contents

- **remarks: (optional).** *Type:* Dictionary.
 - **info (optional).** *Type:* List of strings.
Example: “Text1”,”Text2”.
 A list of information strings concerning the analysis run.
 - **warning (optional).** *Type:* List of strings.
Example: “Text1”,”Text2”.
 A list of warning strings concerning the analysis run.
- **overview.** *Type:* Dictionary.
 - **analysis_engine.** *Type:* String.
Example: “LLama - OSX”.
 Name of the analysis engine used for generating the result.
 - **analysis_engine_version.** *Type:* String.
Example: “1.2.4”.
 Version of the analysis engine used for generating the result.
 - **analysis_start.** *Type:* Date-Time.
Example: “2013-10-05 14:21:01.928894”.
 Start timestamp of the analysis run.
 - **analysis_end.** *Type:* Date-Time.
Example: “2013-10-05 14:22:02.935794”.

End timestamp of the analysis run.

- **analysis_subjects.** *Type:* List of analysis subjects; see *OSX Analysis Subject Format*.

A list of programs monitored during the analysis run.

- **analysis_metadata: (optional).** *Type:* List of analysis metadata; see *OSX Analysis Metadata Format*.

A list of artifacts generated during the analysis run. See *get_result_artifact()* for retrieving this metadata.

- **url_summary: (optional).** *Type:* List of strings.

Example: “http://www.example1.com”, “http://www.example2.com”.

Network summary of contacted URLs during analysis run.

3.4.1 OSX Analysis Subject Format

The analysis engine will monitor all analysis subjects, such as the originally started program and all child processes or processes that a monitored program interacts with, and then list any security relevant data.

This type extends the `format_ll_osx_analysis_subject` type with additional information on OSX analysis subjects.

Analysis subject contents

- **overview.** *Type:* Dictionary.

Overview of the analysis subject. In addition to the base format contents, the following elements are extracted:

- **process.** *Type:* Process; see *OSX Process*.

Information on the OSX process.

- **ext_info: (optional).** *Type:* File-Info; see *Static File Information*.

Static information on the process image.

- **console_output: (optional).** *Type:* Dictionary.

Console output of the program.

- **stdout: (optional).** *Type:* String.

Example: “text written to stdout”.

Program output written to default console.

- **stderr: (optional).** *Type:* String.

Example: “text written to error console”.

Program output written to error console.

- **opened_windows: (optional).** *Type:* List of GUI-windows; see below.

A list of GUI windows opened by the analysis subject.

- **title.** *Type:* String.

Example: “Mac Viewer”.

Window title content.

- **text:** *Type:* String.
Example: “FolderView”.
Window text content.
- **loaded_libraries: (optional).** *Type:* List of libraries; see below.
List of library files loaded by the analysis subject.
 - **filename:** *Type:* String.
Example: “/tmp/foo.so”.
Path to the library loaded by the analysis subject.
- **file_reads: (optional).** *Type:* List of files; see below.
A list of files read by the analysis subject
 - **filename.** *Type:* String.
Example: “test”.
A file name. Could be absolute or relative path.
 - **abs_path: (optional).** *Type:* String.
Example: “/tmp/test”.
An absolute path the file.
 - **ext_info: (optional).** *Type:* File-Info; see *Static File Information*.
Static file information.
- **file_writes: (optional).** *Type:* List of files; see *file_reads*.
A list of files written by the analysis subject.
- **file_deletes: (optional).** *Type:* List of files; see *file_reads*.
A list of files deleted by the analysis subject.
- **file_searches: (optional).** *Type:* List of strings.
A list of files searched for by the analysis subject.
- **process_interactions: (optional).** *Type:* List of process-interactions; see below.
A list of processes the analysis subject interacts with.
In addition to the fields of type *OSX Process*, each element contains the operation(s) performed:
 - **operations:** *Type:* List of strings.
Example: “create_thread”, “write_mem”.
The type of operations performed on the remote process. Possible values are:
 - * “create_process”: Create a process.
 - * “terminate_process”: Terminate a process.
 - * “create_thread”: Create a thread.
 - * “terminate_thread”: Terminate a thread.
- **dns_queries: (optional).** *Type:* List of DNS queries; see *DNS query*.
List of DNS queries done by the analysis subject.

- **network_connections: (optional).** *Type:* List of network connections; see *network connection*.
List of network connections done by the analysis subject using a protocol that is not parsed into a more specific protocol type.
- **http_conversations: (optional).** *Type:* List of HTTP connections; see *HTTP connection*.
List of network connections identified to use the HTTP protocol done by the analysis subject.
- **irc_conversations: (optional).** *Type:* List of IRC connections; see *IRC connection*.
List of network connections identified to use the IRC protocol done by the analysis subject.
- **ftp_conversations: (optional).** *Type:* List of FTP connections; see *FTP connection*.
List of network connections identified to use the FTP protocol done by the analysis subject.
- **smtp_conversations: (optional).** *Type:* List of SMTP connections; see *SMTP connection*.
List of network connections identified to use the SMTP protocol done by the analysis subject.
- **address_scans: (optional).** *Type:* List of network address scans; see *address scan*.
List of network address scans done by the analysis subject.
- **downloaded_files: (optional).** *Type:* List of file-download tuples; see below.
List of files that were downloaded using the OSX file-download API functions. Each element is a tuple of file-origin URL and a File element (see *file_reads*).
Note: This list does not contain files downloaded using other mechanisms or protocol (such as HTTP). Those are listed in the corresponding network section.
- **frequent_api_calls: (optional).** *Type:* List of the frequent API calls; see below.
A list of the frequent API calls.
 - **name: (optional).** *Type:* String.
Example: “open”.
A name of the API function which was frequently called.
 - **count: (optional).** *Type:* String.
Example: 31440.
A number of times the API function was called.
 - **pid: (optional).** *Type:* Integer.
Example: 145.
OSX process identifier (TID) of the calling process.
 - **tid: (optional).** *Type:* Integer.
Example: 167.
OSX thread identifier (TID) of the calling thread.
- **yara_signatures: (optional).** *Type:* List of the Yara signatures; see below.
A list of the Yara signatures which hit on the analysis subject.
 - **name: (optional).** *Type:* String.
Example: “apt_osx_generic_imageStego”.
A name of the Yara signatures.

- **score: (optional).** *Type:* Integer.

Example: 75.

A score which defines how dangerous the analysis subject according to the Yara signature. Possible range from 0 (benign) to 100 (malicious).

- **internal: (optional).** *Type:* Boolean.

If true the signature is only for an internal usage.

3.4.2 OSX Process

Information on an OSX process.

OSX process contents

- **process_id.** *Type:* String.

Example: “1376”.

OSX process identifier (PID).

- **executable: (optional).** *Type:* File; see *file_reads*.

Process image information.

- **arguments: (optional).** *Type:* String.

Example: “/test arg1 arg2”.

Full command line used to start the analysis subject.

- **analysis_subject_id: (optional).** *Type:* Integer.

Example: 2.

Identifier of the analysis subject within the analysis report if the process belongs to an analysis subject monitored in the analysis run.

3.4.3 OSX Analysis Metadata Format

During the analysis run, the analysis engine extracts the metadata that is available for download. This type extends the *Analysis Metadata Format* type.

Metadata contents

- **file: (optional).** *Type:* File; see *file_reads*.

Name of the file generated during the analysis run. Applies to *metadata_type* “generated_file”.

3.5 Report Format *ll-win-timeline-based*

This analysis report format applies to a dynamic analysis run on a Microsoft Windows platform. Differently from *Report Format ll-int-win*, this report focuses on the timeline of different actions and exposes behavior as a series of events associated with the timestamp when each event was observed.

In addition to the report fields that all report formats share (see *Analysis Report Format*), the report contains a number of different fields with details about the analysis run.

Report contents

- **remarks: (optional).** *Type:* Dictionary.
 - **info (optional).** *Type:* List of strings.
Example: “Text1”,”Text2”.
A list of information strings concerning the analysis run.
 - **warning (optional).** *Type:* List of strings.
Example: “Text1”,”Text2”.
A list of warning strings concerning the analysis run.
- **overview.** *Type:* Dictionary.
 - **analysis_engine.** *Type:* String.
Example: “LLama - WindowsXP”.
Name of the analysis engine used for generating the result.
 - **analysis_engine_version.** *Type:* String.
Example: “1.2.4”.
Version of the analysis engine used for generating the result.
 - **analysis_start.** *Type:* Date-Time.
Example: “2013-10-05 14:21:01.928894”.
Start timestamp of the analysis run.
 - **analysis_end.** *Type:* Date-Time.
Example: “2013-10-05 14:22:02.935794”.
End timestamp of the analysis run.
- **analysis_subjects.** *Type:* List of analysis subjects; see *Windows Analysis Subject Format*.
A list of programs monitored during the analysis run.
- **files.** *Type:* List of files; see below.
 - **file_id.** *Type:* Integer.
Example: “1”.
ID used to identify the file.
 - **filename.** *Type:* String.
Example: “desktop.ini”.
A file name. Could be absolute or relative path.
 - **abs_path: (optional).** *Type:* String.
Example: “C:\Users\desktop.ini”.
An absolute file path.
 - **ext_info: (optional).** *Type:* File-Info; see *Static File Information*.
Static file information.

- **analysis_metadata: (optional).** *Type:* List of analysis metadata; see *Windows Analysis Metadata Format*.
A list of artifacts generated during the analysis run. See `get_result_artifact()` for retrieving this metadata.
- **randomized_registry_values: (optional).** *Type:* List of registry keys; see *registry_reads*.
A list of Microsoft Windows Registry values that the analysis engine randomized during the analysis run to avoid detection by the analysis subject. For the format of each value, refer to *registry_reads*.
- **url_summary: (optional).** *Type:* List of strings.
Example: “http://www.example1.com”,”http://www.example2.com”.
Network summary of contacted URLs during analysis run.

3.6 PE Stats information

The analysis engine extracts statistic file information for most PE files manipulated during the analysis run.

PE Stats information contents

- **size.** *Type:* Integer.
Example: 1.
Size of the PE.
- **histogram.** *Type:* List.
Example: “[{‘byte’: 0, frequency: 1.0}]”.
A list with dicts representing the histogram of byte frequency in the PE.
- **entropy.** *Type:* Floating-point number.
Example: 1.5.
An entropy of the PE data.
- **average.** *Type:* Floating-point number.
Example: 1.5.
Average of all byte values in the PE file.
- **variance.** *Type:* Floating-point number.
Variance of the byte values in the PE file.
- **autocorrelation_1.** *Type:* Floating-point number.
Autocorrelation (lag=1) of byte values in the PE data.
- **block_average.** *Type:* List.
A list with the average of byte values in data blocks of 1024 bytes.

3.7 PE Resource Stats information

The analysis engine extracts statistic information about the resources in a PE file for most PE files manipulated during the analysis run.

PE Resource Stats information contents

- **name:** *Type:* String.
Example: “RT_MANIFEST”.
The name of the PE Resource.
- **lang.** *Type:* String.
Example: “LANG_ENGLISH”.
Primary language identifier for the resource.
- **sublang:.** *Type:* String.
Example: “SUBLANG_ENGLISH_US”.
Sublanguage identifier for the resource.
- **stats:.** *Type:* PE Stats; see *PE Stats information*.
PE Resource statistic information.

3.7.1 Windows Analysis Subject Format

The analysis engine will monitor all analysis subjects, such as the originally started program and all child processes or processes that a monitored program interacts with, and then list any security relevant data.

This type extends the *Analysis Subject Format* type with additional information on Windows analysis subjects.

Analysis subject contents

- **overview.** *Type:* Dictionary.
Overview of the analysis subject. In addition to the base format contents, the following elements are extracted:
 - **process.** *Type:* Process; see *Windows Process*.
Information on the Windows process.
 - **ext_info: (optional).** *Type:* File-Info; see *Static File Information*.
Static information on the process image.
- **console_output: (optional).** *Type:* Dictionary.
Console output of the program.
 - **stdout: (optional).** *Type:* String.
Example: “text written to stdout”.
Program output written to default console.
 - **stderr: (optional).** *Type:* String.
Example: “text written to error console”.
Program output written to error console.
- **opened_windows: (optional).** *Type:* List of GUI-windows; see below.
A list of GUI windows opened by the analysis subject.
 - **title:** *Type:* String.
Example: “Documents and Settings”.
Window title content.

- **text:** *Type:* String.

Example: “FolderView”.

Window text content.

- **loaded_libraries: (optional).** *Type:* List of libraries; see below.

List of library files loaded by the analysis subject.

- **filename:** *Type:* String.

Example: “C:\windows\syswow64\ole32.dll”.

Path to the library loaded by the analysis subject.

- **actions:** *Type:* List of actions; see below.

List of actions performed by the analysis subject with timeline information; see below.

- **id: (optional).** *type:* integer.

example: 1.

unique id of the action within the analysis report.

- **action_name:** *Type:* String.

Example: “FileWrite”.

Name of the action performed by the analysis_subject

- **action_type: (optional).** *Type:* String

Example: “write”.

Define the type of action that is performed on the resource.

- **resource_type: (optional).** *Type:* String

Example: “file_resource”.

Define the type of resource (if any) associated with this action.

- **timestamp:** *Type:* Date-Time.

Example: “2013-10-05 14:21:01.928894”.

Timestamp when the action happened.

- **thread_id: (optional)** *Type:* Integer.

Example: 1020.

Thread ID (within the analysis subject) that executed the action.

- **stack_depth: (optional)** *Type:* Integer.

Example: 2.

Inform how many calls we are away from the first call

- **last_timestamp: (optional)** *Type:* Date-Time.

Example: “2013-10-05 14:21:01.928894”.

Timestamp when the action was last repeated.

- **resource: (optional).** *Type:* Resource; see *Action Resources*.
Information on action resource. Name of the resource will match the name passed in resource_type.
- **http_conversations: (optional).** *Type:* List of HTTP connections; see *HTTP connection*.
List of network connections identified to use the HTTP protocol done by the analysis subject.
- **irc_conversations: (optional).** *Type:* List of IRC connections; see *IRC connection*.
List of network connections identified to use the IRC protocol done by the analysis subject.
- **ftp_conversations: (optional).** *Type:* List of FTP connections; see *FTP connection*.
List of network connections identified to use the FTP protocol done by the analysis subject.
- **smtp_conversations: (optional).** *Type:* List of SMTP connections; see *SMTP connection*.
List of network connections identified to use the SMTP protocol done by the analysis subject.
- **address_scans: (optional).** *Type:* List of network address scans; see *address scan*.
List of network address scans done by the analysis subject.
- **downloaded_files: (optional).** *Type:* List of file-download tuples; see below.
List of files that were downloaded using the Microsoft Windows file-download API functions. Each element is a tuple of file-origin URL and a File element (see *file_reads*).
Note: This list does not contain files downloaded using other mechanisms or protocol (such as HTTP). Those are listed in the corresponding network section.
- **pe_images: (optional).** *Type:* List of PE images; see below.
A list of PE images found in the memory of the analysis subject.
 - **image.** *Type:* PE image; see *Portable Executable Image*.
Process image information extracted when included in the analysis.
 - **image_diff: (optional).** *Type:* PE image; see *Portable Executable Image*.
Process image information extracted at program termination or analysis end.
- **memory_blocks: (optional).** *Type:* List of memory-blocks; see below.
A list of allocated memory regions found in the memory of the analysis subject.
 - **name: (optional).** *Type:* String.
Example: “mem_b67f3190f04083ac1e0189307f4d64d4”.
A name of the memory block. Format: mem_<md5>.
 - **size: (optional).** *Type:* Integer.
A size of the memory block in bytes.
 - **start_va: (optional).** *Type:* Integer.
A VA to where the memory block starts.
 - **end_va: (optional).** *Type:* Integer.
A VA to where the memory block ends.
 - **access: (optional).** *Type:* Hexadecimal string.
A set of flags that indicate the memory block’s attributes (such as code/data, readable, or writable).

- **number_of_executed_pages: (optional).** *Type:* Integer.
A number of executed pages in the memory block.
- **executed_pages: (optional).** *Type:* List of integers.
A list of VA of executed pages.
- **dist_bytes_vector: (optional).** *Type:* Hexadecimal string.
An internal field used by an analysis engine.
- **average_bytes: (optional).** *Type:* Hexadecimal string.
An internal field used by an analysis engine.
- **autocorrelation: (optional).** *Type:* Floating-point number.
An autocorrelation of the memory block data.
- **entropy: (optional).** *Type:* Floating-point number.
An entropy of the memory block data.
- **embedded_pe_header.** *Type:* Boolean.
True if we recognized a PE image header in the memory block.
- **number_of_strings: (optional).** *Type:* Integer.
A number of found strings in the memory block.
- **strings: (optional).** *Type:* List of strings.
Example: “SSP3FR.DLL”,”ROOF”,”Help Button”.
A list of strings found in the memory block.
- **md5.** *Type:* Hexadecimal string.
A md5 hash of a content.
- **strings_lists: (optional).** *Type:* List of named strings-lists; see below.
A list of named strings-lists. The name identifies a type of the strings in the list.
 - **name: (optional).** *Type:* String.
Example: “heap_strings”.
A name of the string list.
 - **strings: (optional).** *Type:* List of strings.
Example: “ProgramData=C:\ProgramData”,”NUMBER_OF_PROCESSORS=1”,”ncacn_ip_tcp”.
A list of strings.
- **patched_sleeps: (optional).** *Type:* List of patched sleep values; see below.
A list of patched sleep values. It is an anti-evasion technique which changes a waiting period (if it is too long) for a sleep function and timers.
 - **count: (optional).** *Type:* Integer.
Example: 1.
A number of times the sleep function was called.

- **new_value: (optional).** *Type:* Integer.

Example: 5.

A new value of the waiting period in seconds.

- **old_value: (optional).** *Type:* Integer.

Example: 3600.

A old value of the waiting period in seconds.

- **frequent_api_calls: (optional).** *Type:* List of the frequent API calls; see below.

A list of the frequent API calls.

- **name: (optional).** *Type:* String.

Example: “NtOpenThreadToken”.

A name of the API function which was frequently called.

- **count: (optional).** *Type:* String.

Example: 31440.

A number of times the API function was called.

- **pid: (optional).** *Type:* Integer.

Example: 145.

Windows process identifier (TID) of the calling process.

- **tid: (optional).** *Type:* Integer.

Example: 167.

Windows thread identifier (TID) of the calling thread.

- **yara_signatures: (optional).** *Type:* List of the Yara signatures; see below.

A list of the Yara signatures which hit on the analysis subject.

- **name: (optional).** *Type:* String.

Example: “apt_win_generic_imageStego”.

A name of the Yara signatures.

- **score: (optional).** *Type:* Integer.

Example: 75.

A score which defines how dangerous the analysis subject according to the Yara signature. Possible range from 0 (benign) to 100 (malicious).

- **internal: (optional).** *Type:* Boolean.

If true the signature is only for an internal usage.

- **flirt_signatures: (optional).** *Type:* List of the Yara signatures; see below.

A list of the flirt signatures which hit on the analysis subject. The flirt signatures recognize a known functions.

- **name: (optional).** *Type:* String.

Example: “__ascii_strnicmp”

Name of the signature.

- **keyboard_capture: (optional).** *Type:* List of keyboard actions captured; see below.

A list of keyboard actions sent during the analysis which were seen on the behavior.

- **word: (optional).** *Type:* String.

Example: “8989-8408-5161-4765”

Word captured by sample during analysis.

- **word_type: (optional).** *Type:* String.

Example: “Credit Card”

Type of word that was captured.

3.7.2 Action Resources

Information about action resource types

Action resource contents

- **file_resource:**

Type: List with file information attributes.

File resource information

- **file_id.** *Type:* Integer

Example: 1.

An identifier for the file resource used, see *files*.

- **status. (optional)** *Type:* String

Example: “STATUS_UNKNOWN”.

Return value for a file operation.

- **file_attributes. (optional)** *Type:* List of Strings

Example: “FILE_ATTRIBUTE_ARCHIVE”, “FILE_ATTRIBUTE_HIDDEN”.

Attributes of the file.

- **iostatus. (optional)** *Type:* List of strings

Example: “FILE_CREATED”, “FILE_OPENED”

Completion status of the file operation.

- **pe_resources. (optional)** *Type:* List.

List of resources in the PE, see *pe-resources*.

- **pe_overlay_stats. (optional)** *Type:* Dictionary

Statistics about the file (if it is a PE), see *pe-stats*.

- **disposition. (optional)** *Type:* List of strings

Example: “FILE_OVERWRITE_IF”, “FILE_OPEN”.

How to proceed when opening the file.

- **options. (optional)** *Type:* List of strings

Example: “FILE_NON_DIRECTORY_FILE”, “FILE_SYNCHRONOUS_IO_NONALERT”.
Creation options for the file.

- **registry_resource:** *Type:* List with registry key information.

A registry key used by the analysis subject.

- **key.** *Type:* String.

Example: “HKLM\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\DRIVERS32”.
A registry key.

- **value: (optional).** *Type:* String.

Example: “wave9”.
A registry value.

- **data: (optional).** *Type:* String or Integer.

Example: 1, “mso.dll”.
A data of registry value.

- **mutex_resource:** *Type:* List with mutex information.

Mutex information from a mutex used by the analysis subject.

- **mutex_name:** *Type:* String.

Example: “Mutex1”.
Name of the mutex synchronization object.

- **error_resource:** *Type:* List with system error information.

An exceptions raised by the analysis subject.

- **addr: (optional).** *Type:* Hexadecimal number.

Example: 0x7c832297.
Instruction address raising the exception.

- **code: (optional).** *Type:* Hexadecimal number.

Example: 0xc0000005.
Microsoft Windows exception code.

- **name: (optional).** *Type:* String.

Example: “STATUS_ACCESS_VIOLATION”.
Microsoft Windows exception name.

- **exception_name: (optional).** *Type:* String.

Example: “Exception 0xc0000005 (STATUS_ACCESS_VIOLATION) at 0x7c832297 (subject_id: 2)”.
Full exception information.

- **exception_count: (optional).** *Type:* Integer.

Example: 1.

Number of exception occurrences.

- **errors: (optional).** *Type:* List.

Example: “SEM_NOOPENFILEERRORBOX”, “SEM_FAILCRITICALERRORS”.

Microsoft Windows process error modes.

- **process_resource:** *Type:* Process information

A Process used by the analysis subject; see *process*.

- **hook_resource:** *Type:* List with hook information.

A Hook set by the analysis subject.

- **hooks:**

Type: List of hooks set in the system

Example: “WH_KEYBOARD_LL”, “WH_KEYBOARD”, “WH_MOUSE”.

A list of strings containing the hooks set in system.

- **service_resource:** *Type:* List with service information.

An service used by the analysis subject.

- **service_file: (optional).** *Type:* Integer

Example: 1.

File ID. see *files*.

- **service_name: (optional).** *Type:* String.

Example: “WNLiserv”

A display name to be used by user interface programs to identify the service.

- **start_type: (optional).** *Type:* Integer.

Example: 0x00000002 (SERVICE_FILE_SYSTEM_DRIVER)

- **change: (optional).** *Type:* String.

A change of the service made by the analysis subject.

- **search_resource:** *Type:* List with file search information.

Search made by the analysis subject.

- **searched_data: (optional).** *Type:* List of data searched

Example: “C:\TEST.DLL”, “D:*”, “C:\WINDOWS”.

A list of strings containing the searched information.

- **network_resource:** *Type:* A network connection see; *network connection*.

Network connection done by the analysis subject using a protocol that is not parsed into a more specific network action type.

- **dns_resource:** *Type:* DNS query; see *DNS query*.

DNS query done by the analysis subject.

- **string_cmp_resource:** *Type:* List with string comparison attributes.

A string comparison.

- **name: (optional).** *Type:* String.
Example: “shlwapi.dll.StrStr_generic”.
A possible name of the string comparison function. Usually defined by a flirt signature *flirt_signatures*.
- **src_string: (optional).** *Type:* String.
Example: “C:\Users\Public\Desktop”.
A comparable string 1.
- **dst_string: (optional).** *Type:* String.
Example: “%SYSTEMROOT%”.
A comparable string 2.
- **src_sources: (optional).** *Type:* List of strings.
Example: “\Registry\Machine\Software\Classes.dot\Icon”, “Command line”.
A list of possible sources (from where this string could be read) for string 1.
- **dst_sources: (optional).** *Type:* List of strings.
Example: “\Registry\Machine\Software\Classes.dot\Icon”, “Command line”.
A list of possible sources (from where this string could be read) for string 2.

3.7.3 Windows Process

Information on a Windows process.

Windows process contents

- **process_id.** *Type:* String.
Example: “1376”.
Windows process identifier (PID).
- **executable: (optional).** *Type:* File; see *file_reads*.
Process image information.
- **arguments: (optional).** *Type:* String.
Example: “C:\subject.exe arg1 arg2”.
Full command line used to start the analysis subject.
- **bitsize: (optional).** *Type:* Integer.
Example: 32.
Process bit-size (32bit or 64bit process).
- **analysis_subject_id: (optional).** *Type:* Integer.
Example: 2.
Identifier of the analysis subject within the analysis report if the process belongs to an analysis subject monitored in the analysis run.

3.8 Report Format *ll-osx-timeline-based*

This analysis report format applies to a dynamic analysis run on a OSX platform. Differently from *Report Format ll-int-osx*, this report focuses on the timeline of different actions and exposes behavior as a series of events associated with the timestamp when each event was observed.

In addition to the report fields that all report formats share (see *Analysis Report Format*), the report contains a number of different fields with details about the analysis run.

Report contents

- **remarks: (optional).** *Type:* Dictionary.
 - **info (optional).** *Type:* List of strings.
Example: “Text1”,”Text2”.
A list of information strings concerning the analysis run.
 - **warning (optional).** *Type:* List of strings.
Example: “Text1”,”Text2”.
A list of warning strings concerning the analysis run.
- **overview.** *Type:* Dictionary.
 - **analysis_engine.** *Type:* String.
Example: “LLama - OSX”.
Name of the analysis engine used for generating the result.
 - **analysis_engine_version.** *Type:* String.
Example: “1.2.4”.
Version of the analysis engine used for generating the result.
 - **analysis_start.** *Type:* Date-Time.
Example: “2013-10-05 14:21:01.928894”.
Start timestamp of the analysis run.
 - **analysis_end.** *Type:* Date-Time.
Example: “2013-10-05 14:22:02.935794”.
End timestamp of the analysis run.
- **analysis_subjects.** *Type:* List of analysis subjects; see *OSX Analysis Subject Format*.
A list of programs monitored during the analysis run.
- **files.** *Type:* List of files; see below.
 - **file_id.** *Type:* Integer.
Example: “1”.
ID used to identify the file.
 - **filename.** *Type:* String.
Example: “foo”.
A file name. Could be absolute or relative path.

- **abs_path: (optional).** *Type:* String.
Example: “/tmp/foo”.
An absolute file path.
- **ext_info: (optional).** *Type:* File-Info; see *Static File Information*.
Static file information.
- **analysis_metadata: (optional).** *Type:* List of analysis metadata; see *OSX Analysis Metadata Format*.
A list of artifacts generated during the analysis run. See *get_result_artifact ()* for retrieving this metadata.
- **url_summary: (optional).** *Type:* List of strings.
Example: “http://www.example1.com”, “http://www.example2.com”.
Network summary of contacted URLs during analysis run.

3.8.1 OSX Analysis Subject Format

The analysis engine will monitor all analysis subjects, such as the originally started program and all child processes or processes that a monitored program interacts with, and then list any security relevant data.

This type extends the *Analysis Subject Format* type with additional information on OSX analysis subjects.

Analysis subject contents

- **overview.** *Type:* Dictionary.
Overview of the analysis subject. In addition to the base format contents, the following elements are extracted:
 - **process.** *Type:* Process; see *OSX Process*.
Information on the OSX process.
 - **ext_info: (optional).** *Type:* File-Info; see *Static File Information*.
Static information on the process image.
- **console_output: (optional).** *Type:* Dictionary.
Console output of the program.
 - **stdout: (optional).** *Type:* String.
Example: “text written to stdout”.
Program output written to default console.
 - **stderr: (optional).** *Type:* String.
Example: “text written to error console”.
Program output written to error console.
- **opened_windows: (optional).** *Type:* List of GUI-windows; see below.
A list of GUI windows opened by the analysis subject.
 - **title:** *Type:* String.
Example: “Mac Viewer”.
Window title content.

- **text:** *Type:* String.

Example: “FolderView”.

Window text content.

- **loaded_libraries:** (optional). *Type:* List of libraries; see below.

List of library files loaded by the analysis subject.

- **filename:** *Type:* String.

Example: “/tmp/bar.so”.

Path to the library loaded by the analysis subject.

- **actions:** *Type:* List of actions; see below.

List of actions performed by the analysis subject with timeline information; see below.

- **id:** (optional). *type:* integer.

example: 1.

unique id of the action within the analysis report.

- **action_name:** *Type:* String.

Example: “FileWrite”.

Name of the action performed by the analysis_subject

- **action_type:** (optional). *Type:* String

Example: “write”.

Define the type of action that is performed on the resource.

- **resource_type:** (optional). *Type:* String

Example: “file_resource”.

Define the type of resource (if any) associated with this action.

- **timestamp:** *Type:* Date-Time.

Example: “2013-10-05 14:21:01.928894”.

Timestamp when the action happened.

- **thread_id:** (optional) *Type:* Integer.

Example: 1020.

Thread ID (within the analysis subject) that executed the action.

- **stack_depth:** (optional) *Type:* Integer.

Example: 2.

Inform how many calls we are away from the first call

- **last_timestamp:** (optional) *Type:* Date-Time.

Example: “2013-10-05 14:21:01.928894”.

Timestamp when the action was last repeated.

- **resource: (optional).** *Type:* Resource; see *Action Resources*.

Information on action resource. Name of the resource will match the name passed in resource_type.

- **http_conversations: (optional).** *Type:* List of HTTP connections; see *HTTP connection*.

List of network connections identified to use the HTTP protocol done by the analysis subject.

- **irc_conversations: (optional).** *Type:* List of IRC connections; see *IRC connection*.

List of network connections identified to use the IRC protocol done by the analysis subject.

- **ftp_conversations: (optional).** *Type:* List of FTP connections; see *FTP connection*.

List of network connections identified to use the FTP protocol done by the analysis subject.

- **smtp_conversations: (optional).** *Type:* List of SMTP connections; see *SMTP connection*.

List of network connections identified to use the SMTP protocol done by the analysis subject.

- **address_scans: (optional).** *Type:* List of network address scans; see *address scan*.

List of network address scans done by the analysis subject.

- **downloaded_files: (optional).** *Type:* List of file-download tuples; see below.

List of files that were downloaded using the OSX file-download API functions. Each element is a tuple of file-origin URL and a File element (see *file_reads*).

Note: This list does not contain files downloaded using other mechanisms or protocol (such as HTTP). Those are listed in the corresponding network section.

- **frequent_api_calls: (optional).** *Type:* List of the frequent API calls; see below.

A list of the frequent API calls.

- **name: (optional).** *Type:* String.

Example: “open”.

A name of the API function which was frequently called.

- **count: (optional).** *Type:* String.

Example: 31440.

A number of times the API function was called.

- **pid: (optional).** *Type:* Integer.

Example: 145.

OSX process identifier (PID) of the calling process.

- **tid: (optional).** *Type:* Integer.

Example: 167.

OSX thread identifier (TID) of the calling thread.

- **yara_signatures: (optional).** *Type:* List of the Yara signatures; see below.

A list of the Yara signatures which hit on the analysis subject.

- **name: (optional).** *Type:* String.

Example: “apt_osx_generic_imageStego”.

A name of the Yara signatures.

- **score: (optional).** *Type:* Integer.

Example: 75.

A score which defines how dangerous the analysis subject according to the Yara signature. Possible range from 0 (benign) to 100 (malicious).

- **internal: (optional).** *Type:* Boolean.

If true the signature is only for an internal usage.

3.8.2 Action Resources

Information about action resource types

Action resource contents

- **file_resource:**

Type: List with file information attributes.

File resource information

- **file_id.** *Type:* Integer

Example: 1.

An identifier for the file resource used, see *files*.

- **process_resource:** *Type:* Process information

A Process used by the analysis subject; see *process*.

- **search_resource:** *Type:* List with file search information.

Search made by the analysis subject.

- **searched_data: (optional).** *Type:* List of data searched

Example: “/tmp/foo”, “/tmp/*.?”.

A list of strings containing the searched information.

- **network_resource:** *Type:* A network connection see; *network connection*.

Network connection done by the analysis subject using a protocol that is not parsed into a more specific network action type.

- **dns_resource:** *Type:* DNS query; see *DNS query*.

DNS query done by the analysis subject.

3.8.3 OSX Process

Information on an OSX process.

OSX process contents

- **process_id.** *Type:* String.

Example: “1376”.

OSX process identifier (PID).

- **executable: (optional).** *Type:* File; see *file_reads*.
Process image information.
- **arguments: (optional).** *Type:* String.
Example: “/tmp/test arg1 arg2”.
Full command line used to start the analysis subject.
- **analysis_subject_id: (optional).** *Type:* Integer.
Example: 2.
Identifier of the analysis subject within the analysis report if the process belongs to an analysis subject monitored in the analysis run.

3.9 Report Format *ll-int-win-doc*

This analysis report format refers to a dynamic analysis run of opening a document using Microsoft Office on a Microsoft Windows platform.

In addition to the report fields of *Report Format ll-int-win*, the report contains additional information about the analyzed document.

Report contents

- **static_analysis. (optional).** *Type:* Dictionary.
 - **document_content. (optional).** *Type:* String.
Example: “This is a document analyzed by Microsoft Windows”.
A text portion extracted from the analyzed document

3.10 Report Format *ll-int-apk*

This analysis report format applies to a dynamic analysis run on a Android platform (Deprecated). This type extends the *Analysis Report Format* type.

Report contents

- **api_level: (optional).** *Type:* Integer.
Example: 3.
Android API level used by this analysis run.
- **analysis_subjects.** *Type:* List of analysis subjects; see *Android Analysis Subject Format*.
A list of programs monitored during the analysis run.

3.10.1 Android Analysis Subject Format

The analysis engine will monitor all analysis subjects, such as the originally started program and all child processes or processes that a monitored program interacts with, and then list any security relevant data.

This type extends the *Analysis Subject Format* type with additional information on Android analysis subjects.

Analysis subject contents

- **overview.** *Type:* Dictionary.
 Overview of the analysis subject. In addition to the base format contents, the following elements are extracted:
 - **program_name.** *Type:* String.
 Name of the analyzed program.
- **valid_manifest: (optional).** *Type:* Boolean.
 True if the Android package has a valid manifest.
- **valid_zipfile: (optional).** *Type:* Boolean.
 True if the Android package is a valid ZIP file.
- **valid_androguard_zipfile: (optional).** *Type:* Boolean.
 True if the Android package is a valid ZIP file that can be processed by AndroGuards analysis tool.
- **uses_native_code: (optional).** *Type:* Boolean.
 True if the Android package makes use of native code.
- **uses_dynamic_code: (optional).** *Type:* Boolean.
 True if the Android package makes use of dynamic code.
- **uses_reflection: (optional).** *Type:* Boolean.
 True if the Android package makes use of reflection.
- **uses_crypto: (optional).** *Type:* Boolean.
 True if the Android package makes use of cryptographic functionality.
- **certificate: (optional).** *Type:* Dictionary.
 The certificate included in the Android package.
 - **valid_from: (optional).** *Type:* String.
Example: “Sat Jan 01 00:00:00 GMT 2011”.
 The certificate validity start-date as extracted from the certificate.
 - **valid_until: (optional).** *Type:* String.
Example: “Sat Dec 23 01:23:45 GMT 2045”.
 The certificate validity end-date as extracted from the certificate.
 - **owner: (optional).** *Type:* String.
Example: “CN=John Doe, OU=android, O=My Apps, L=Dallas, ST=TX, C=US”.
 The certificate owner as extracted from the certificate.
 - **issuer: (optional).** *Type:* String.
Example: “CN=John Doe, OU=android, O=My Apps, L=Dallas, ST=TX, C=US”.
 The certificate issuer as extracted from the certificate.
 - **serial_number: (optional).** *Type:* String.
Example: “4d1a9bb1”.
 The certificate serial-number as extracted from the certificate.

- **md5: (optional).** *Type:* Hexadecimal string/hash.
Example: “AA:CC:12:FE:BB:C1:87:3E:08:44:DF:12:D4:6F:39:43”.
The certificate MD5 hash as extracted from the certificate.
- **sha1: (optional).** *Type:* Hexadecimal string/hash.
Example: “AA:CC:12:FE:BB:C1:87:3E:08:44:DF:12:D4:6F:39:43:BB:C4:46:F9”.
The certificate SHA1 hash as extracted from the certificate.
- **required_features: (optional).** *Type:* List of strings.
Example: “android.hardware.touchscreen”, “android.hardware.location.gps”.
A list of Android features (by name) required by the Android application.
- **permissions: (optional).** *Type:* List of permissions; see below.
Permissions required/used by the Android application.
 - **permission: (optional).** *Type:* String.
Example: “android.permission.WRITE_EXTERNAL_STORAGE”.
Name of the permission.
 - **calls: (optional).** *Type:* List of calls; see [Android Function Call](#).
Function calls inside the Android application that indicate the use of the given permission.
- **activities: (optional).** *Type:* List of activities; see below.
Activities supported by the Android application.
 - **name: (optional).** *Type:* String.
Example: “MainLogin”.
Name of the activity.
 - **intent_filters: (optional).** *Type:* List of intent-filters; see [Android Intent Filter](#).
Intent-filters registered on the activity.
- **broadcast_receivers: (optional).** *Type:* List of broadcast-receivers; see below.
Broadcast-receivers supported by the Android application.
 - **name: (optional).** *Type:* String.
Example: “com.amazon.inapp.purchasing.ResponseReceiver”.
Name of the broadcast-receiver.
 - **intent_filters: (optional).** *Type:* List of intent-filters; see [Android Intent Filter](#).
Intent-filters registered on the broadcast-receiver.
- **service_creates: (optional).** *Type:* List of services. See below.
A list of services created by the analysis subject.
 - **service_name: (optional).** *Type:* String.
Example: “com.movend.market_billing.BillingService”.
The name of the service.

- **intent_filters: (optional).** *Type:* List of intent-filters; see *Android Intent Filter*.

Intent-filters registered on the service.

- **service_starts: (optional).** *Type:* List of services. See *service_creates*.

A list of services started by the analysis subject.

- **file_reads: (optional).** *Type:* List of files. See below.

A list of files read by the analysis subject.

- **filename: (optional).** *Type:* String.

Example: “/data/data/com.android.mms/shared_prefs/com.android.mms_preferences.xml”

The name of the file.

- **file_writes: (optional).** *Type:* List of files. See *file_reads*.

A list of files written by the analysis subject.

- **file_deletes: (optional).** *Type:* List of files. See *file_reads*.

A list of deleted written by the analysis subject.

- **file_leaks: (optional).** *Type:* List of files. See *file_reads*.

A list of files read and leaked to an external party by the analysis subject.

- **urls: (optional).** *Type:* List of strings.

Example: “http://test.com”, “https://test.org”.

A list of URLs embedded inside the Android application.

- **dns_queries: (optional).** *Type:* List of DNS queries; see *DNS query*.

List of DNS queries done by the analysis subject.

- **network_connections: (optional).** *Type:* List of network connections; see *network connection*.

List of network connections done by the analysis subject using a protocol that is not parsed into a more specific protocol type.

- **http_conversations: (optional).** *Type:* List of HTTP connections; see *HTTP connection*.

List of network connections identified to use the HTTP protocol done by the analysis subject.

3.10.2 Android Function Call

Information about a function call.

Function call contents

- **call_site: (optional).** *Type:* String.

Location (module/function name) of the call.

- **called_function: (optional).** *Type:* String.

Called function (module/function name) that requires the given permission.

- **call_site_object: (optional).**

call_site parsed to object. Added for compatibility.

- **function_name: (optional).** *Type:* String.
Example: “disableKeyguard”.
The call function name.
 - **arguments: (optional).** *Type:* List of argument dictionaries.
Example: “[{‘type’: ‘int’}, {‘type’: ‘java.lang.String’}]”.
Function arguments (types only).
 - **return_type: (optional).** *Type:* String.
Example: ‘java/lang/String’.
Function return type.
 - **class_name: (optional).** *Type:* String.
Example: “rnasdvs.addB”.
Class which function belongs to.
 - **module_name: (optional).** *Type:* String.
Example: “com.tartiap.lnnhdatu”.
Module for class and function.
- **called_function_object:** (optional).
called_function parsed to object. Added for compatibility.
 - **function_name: (optional).** *Type:* String.
Example: “disableKeyguard”.
The call function name.
 - **arguments: (optional).** *Type:* List of argument dictionaries.
Example: “[{‘type’: ‘int’}, {‘type’: ‘java.lang.String’}]”.
Function arguments (types only).
 - **return_type: (optional).** *Type:* String.
Example: ‘java/lang/String’.
Function return type.
 - **class_name: (optional).** *Type:* String.
Example: “rnasdvs.addB”.
Class which function belongs to.
 - **module_name: (optional).** *Type:* String.
Example: “com.tartiap.lnnhdatu”.
Module for class and function.

3.10.3 Android Intent Filter

Intent-filters registered on the Android activities, broadcast-receivers or services.

Intent filter contents

- **action: (optional).** *Type:* String.
Example: “android.intent.action.MAIN”.
Intent-filter action.
- **category: (optional).** *Type:* String.
Example: “android.intent.category.LAUNCHER”.
Intent-filter category.

3.11 Report Format *ll-int-archive*

This analysis report format applies to a run of the archive extractor. This type extends the *Analysis Report Format* type.

Report contents

- **extracted_files.** *Type:* List of extracted files. See *Extracted Files*.
A list of files extracted from the archive.
- **full_archive_analysis.** *Type:* List of hexadecimal strings.
Example: [“7065a3ba0c729ad5981a1e1072df710d”].
A list of unique identifiers for analysis submissions that are associated with analyzing the archive as a whole. This value can be used to obtain a report for the corresponding child task (see `get_result()`).

3.11.1 Extracted Files

A file that was extracted from the submitted archive.

Extracted file contents

- **md5** *Type:* Hexadecimal string.
Example: “748cb82987899a164c2f6e7985ffec5”.
An md5 hash of the file content.
- **sha1** *Type:* Hexadecimal string.
Example: “066e791be6fb28063fc643cea658bf70d193b895”.
A sha1 hash of the file content.
- **mime_type: (optional).** *Type:* String.
Example: “application/vnd.android.package-archive”.
The MIME type of the extracted file.

- **task_uuid.** *Type:* Hexadecimal string.

Example: 7065a3ba0c729ad5981a1e1072df710d.

Unique identifier for the analysis submission of the extracted file. This value can be used to obtain a report for the child task (see `get_result()`).

3.12 Report Format //web

This analysis report format refers to a dynamic analysis run of a web page or of a PDF document.

In addition to the report fields shared by all report formats (see *Analysis Report Format*) the report contains a number of different fields with details specific to the analysis run.

Reports may include fields not described here: they are to be considered as experimental or deprecated and SHOULD be ignored.

Report contents

- **analysis.** *Type:* Dictionary.
 - **applets.** *Type:* Dictionary with applet information; see *Applets Format* for details.
A dictionary giving details about the contents of applets found during the analysis.
 - **artifacts (deprecated).** *Type:* List.
This field is deprecated and should be ignored.
 - **dropped_files.** *Type:* List of dropped files; see *Files Dropped to Disk Format* for details.
The list of files that were dropped to disk during the analysis.
 - **evals.** *Type:* List of dynamically evaluated code; see *Code Format* for details.
A list with details about code that was dynamically evaluated during the analysis, via `eval()`, `setTimeout()`, or similar mechanisms.
 - **exploits.** *Type:* List of exploited vulnerabilities; see *Exploits Format* for details.
A list describing each vulnerability that was found to be exploited during the analysis.
 - **hidden_elements.** *Type:* List of hidden HTML elements; see *Hidden Element Format* for details.
A list with details about hidden HTML elements that cause external resources to be fetched.
 - **network.** *Type:* Dictionary describing the network activity that was recorded during the analysis.
 - * **requests:** *Type:* List of HTTP requests; see *Request Format*.
The list of HTTP requests and responses that were performed during the analysis.
 - **new_functions.** *Type:* List of dynamically evaluated code; see *Code Format* for details.
A list with details about code that was dynamically evaluated during the analysis, via `new Function()` or similar mechanisms.
 - **plugins.** *Type:* List of plugins; see *Plugin Format*.
The list of plugins and ActiveX controls that were loaded during the visit.
 - **processes.** *Type:* List of processes that were unexpectedly spawned during the analysis; see *Processes Unexpectedly Spawned during the Analysis Format*.

- **resources.** *Type:* List of local resources that were accessed during the analysis; see *Local Resource Format*.
The list of local resources that were accessed using the *res://* protocol during the analysis.
- **result.** *Type:* Dictionary.
A dictionary that provides additional information about the analysis results.
 - * **analysis_ended** *Type:* String
Example: “2013-10-03 11:36:42+0000”
Time when the analysis ended in the analysis (UTC timezone).
 - * **classification.** *Type:* String:
Deprecated: this field should be ignored in favor of the top-level score file; see *Contents of response*.
Example: “malicious”
 - * **detector.** *Type:* String
Example: “2.6”
The internal version of the URL and PDF analyzer.
 - * **explanation.** *Type:* String
Deprecated: this field should be ignored in favor of the top-level score file; see *Contents of response*.
Example: “exploits”
- **shellcodes.** *Type:* List of shellcodes; see *Shellcode Format*.
- **statics.** *Type:* List of code that was statically included in a visited page; see *Code Format* for details.
A list with details about code that was statically found during the analysis.
- **strings.** *Type:* List of strings that were found during the analysis; see *String Format* for details.
A list with details about interesting strings that were found during the analysis in the browser’s memory.
- **subject** *Type:* Dictionary describing the analysis subject.
 - * **md5.** *Type:* hexadecimal string.
Example: 6705f99eccedeac20e969bef954c5fb0
MD5 of the input file; not present in URL submissions
 - * **type.** *Type* either “file” or “url”
Example: “url”
 - * **url.** *Type* URL if the engine analyzed a URL submission; null otherwise.
- **text_from_documents.** *Type:* List of textual content extracted from PDF; see *Textual Content Format*.
- **urls_from_documents.** *Type:* List of links extracted from PDF; see *Links Extracted from PDFs Format*.

- **writes.** *Type:* List of code that was dynamically evaluated in a visited page, via *document.write*; see *Code Format* for details.

A list with details about code that was dynamically evaluated via *document.write*.

3.12.1 Exploits Format

A vulnerability that was exploited during the analysis.

- **exploit_id.** *Type:* String.

Example: “81”

The internal unique identifier for this vulnerability.

- **from_url.** *Type:* String.

Example: “<http://evil.example.com/>”.

The URL where the exploit for this vulnerability was found.

- **reference_id.** *Type:* String.

Example: “CVE-2009-0927”

The public vulnerability ID, such as its CVE number.

- **reference_url.** *Type:* String.

Example: “<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-0927>”

A URL where more information about the vulnerability can be found.

3.12.2 Request Format

A request that was issued during the analysis.

- **content_md5.** *Type:* Hexadecimal string.

Example: 6705f99eccedeac20e969bef954c5fb0.

The MD5 hash of the response content.

- **content_sha1.** *Type:* Hexadecimal string.

Example: “c3499c2729730a7f807efb8676a92dcb6f8a3f8f”.

The SHA1 hash of the response content.

- **content_type.** *Type:* String.

Example: “application/octet-stream”.

The content type of the response.

- **ip.** *Type:* String.

Example: “192.0.2.1”.

The IP address of the contacted server; null if the domain name resolution failed.

- **parent_url.** *Type:* String.

Example: “<http://example.com/>”.

The URL that caused the current URL to be fetched; the special value “USER_URL” is reserved for the first URL (which was submitted by the user).

- **relation_type.** *Type:* Integer.

Example: 1

Specifies the mechanism that caused the browser to fetch the content of the URL provided in the *url* field. It can be one of:

- 0 (IFRAME): through an HTML iframe tag
- 1 (SCRIPT_SRC): through an HTML script tag
- 2 (WINDOW_OPEN): by executing window.open in JavaScript
- 3 (FRAME): through an HTML frame tag
- 4 (REDIRECTION): via a redirection (e.g., a 302 response from the server)
- 5 (OTHER): any other method
- 6 (USER): direct user request
- 7 (AJAX): through an Ajax request
- 8 (PLUGIN): via plugin request
- 9 (IMAGE): through an img HTML tag
- 10 (JS): by changing the location in JavaScript (for example, location.setUrl, etc.)
- 11 (LINK): via an HTML link
- 12 (CSS): via some CSS construct
- 13 (REFRESH): by refreshing the page

- **relation_type_str.** *Type:* String.

Example: “USER”

Specifies the mechanism that caused the browser to fetch the content of the URL provided in the *url* field. In comparison to the *relation_type* field, the mechanism is specified as a human-friendly string, rather than as an integer.

- **status.** *Type:* Integer.

Example: 200

The HTTP status code provided by the server.

- **url.** *Type:* String.

Example: “<http://www.example.com>”.

The URL that was requested during the analysis.

3.12.3 Plugin Format

A plugin or ActiveX control that was loaded during the visit of a page.

- **attributes** *Type:* List list of attributes.

For each attribute, the report specifies:

- **name.** *Type:* String.
Example: “propDownloadUrl”.
The name of the attribute.
- **values.** *Type:* List of String.
Example: “http://www.example.com/evil.exe”.
The values assigned to this attribute.
- **classid.** *Type:* String.
Example: “C1B7E532-3ECB-4E9E-BB3A-2951FFE67C61”.
The plugin/ActiveX classid.
- **methods.** *Type:* List of methods calls
For each method, the report specifies:
 - **calls.** *Type:* List of method calls. Each method call consists of a list of strings.
Example: “%45000f”
 - **name.** *Type:* String.
Example: “Collab.getIcon”.
The name of the invoked method.

3.12.4 Shellcode Format

Shellcode extracted from memory during the analysis.

- **from_url.** *Type:* String.
Example: “http://evil.example.com/”.
The URL where the shellcode was found.
- **shellcode_ascii.** *Type:* String.
The shellcode as a printable ASCII string.
- **shellcode_base64.** *Type:* String.
The shellcode base64-encoded.
- **shellcode_hex.** *Type:* Hexadecimal string.
The shellcode as an hexadecimal string.

3.12.5 Textual Content Format

Textual content that has been extracted from PDF files.

- **doc_md5.** *Type:* Hexadecimal string.
Example: “c9d2242bb263603b80916fec27e9f2bb”.
The MD5 hash of the document from where the content was extracted.

- **doc_sha1.** *Type:* Hexadecimal string.
Example: “38a920093773c3e4a7d571f3cd6c5326cadbe5c2”.
The SHA1 hash of the document from where the content was extracted.
- **text.** *Type:* String.
Example: “This is a test PDF file”.
The actual text content extracted from the document.

3.12.6 Links Extracted from PDFs Format

Links that have been extracted from PDF files.

- **child_url_type.** *Type:* String.
Example: “url-in-pdf”.
The type of link that was extracted. Currently, the only value supported is “url-in-pdf”, which indicates that the link was extracted from a PDF document.
- **source_url.** *Type:* String.
Example: “file://ed1fd0c15690007009c4cadcbd677c01/”.
The URL of the document that contains the link.
- **task_uuid (optional).** *Type:* Hexadecimal string.
Example: 7065a3ba0c729ad5981a1e1072df710d.
Unique identifier for the analysis submission of the extracted file. This value can be used to obtain a report for the child task (see `get_result()`).
- **url.** *Type:* String.
Example: “http://example.com/”.
The link’s URL.

3.12.7 Processes Unexpectedly Spawned during the Analysis Format

Processes that have been spawned unexpectedly during the analysis and that are typically associated with exploitation activities.

- **command_line.** *Type:* String.
Example: “\c script.bat”
The command line that was executed.
- **application_name.** *Type:* String.
Example: “C:\Users\ExampleUser\AppData\Local\Temp\52E0.tmp”
The name of the application that was unexpectedly launched.
- **task_uuid (optional).** *Type:* Hexadecimal string.
Example: 7065a3ba0c729ad5981a1e1072df710d.
Unique identifier for the analysis submission of the extracted file. This value can be used to obtain a report for the child task (see `get_result()`).

3.12.8 Files Dropped to Disk Format

Files that have been saved to disk during the analysis.

- **filename.** *Type:* String.
Example: “C:\Users\Johnson\Downloads\nQ30”
The filename of the dropped file.
- **md5:** *Type:* Hexadecimal string.
Example: “e83bbd9d04cf15ee35a2911be221ae3b”.
The MD5 hash of the dropped file.
- **sha1:** *Type:* Hexadecimal string.
Example: “6cf755e7ada47b9bea97dadd65f6140ed1863ca2”.
The SHA1 hash of the dropped file.
- **task_uuid (optional).** *Type:* Hexadecimal string.
Example: 7065a3ba0c729ad5981a1e1072df710d.
Unique identifier for the analysis submission of the dropped file. This value can be used to obtain a report for the child task (see `get_result()`).

3.12.9 Code Format

Code evaluated during the analysis.

- **code.** *Type:* String.
Example: “alert(‘test’)”
The raw code content.
- **code_beautified.** *Type:* String
Example: “alert(‘test’)”
A beautified version of the raw code content; null, if it was not possible to beautify the original code.
- **codecluster_result.** *Type:* Object with the result of the code clustering; see *Codecluster Format* for details.
If the original code is found to be similar to one of the code clusters that we track, this field will contain more details about the matching cluster.
- **media_type.** *Type:* String.
Example: “application/javascript”.
The media type of the code.
- **source_url.** *Type:* String.
Example: “http://example.com”.
The URL of the resource that contains the code.

3.12.10 String Format

Strings found in the browser's memory during the analysis.

- **source_url.** *Type:* String.

Example: "http://example.com".

The URL of the resource being evaluated when the string was found.

- **str_len.** *Type:* Integer.

Example: 42.

The length of the string.

- **str_type.** *Type:* String.

Example: "s".

The type of the string: "s" for a string that was statically found in the content of a visited resource; "d" for a string that was found while dynamically evaluating a visited resource.

- **value.** *Type:* String.

Example: "Test".

The actual string value.

3.12.11 Local Resource Format

A local resource that was accessed via the *res://* protocol.

- **category.** *Type:* String.

Example: "Debugging".

The resource category.

- **path.** *Type:* String.

Example: "res://C:\Program Files (x86)\Fiddler2\Fiddler.exe/#3/#32512"

The resource path.

- **program.** *Type:* String.

Example: "Fiddler".

The program containing the resource.

3.12.12 Hidden Element Format

A hidden element that causes external resources to be fetched.

- **element_type.** *Type:* String.

Example: "iframe".

The element type, for example "iframe" or "frame".

- **resource_url.** *Type:* String.

Example: "http://example.com".

The URL fetched by the hidden element.

- **source_url.** *Type:* String.
Example: “http://example.com”.
The URL containing the hidden element.
- **tag.** *Type:* String.
Example: “<iframe class=’hidden’ src=’http://example.com’></iframe>”.
The hidden element’s code.

3.12.13 Codecluster Format

Information about a code cluster match.

- **description.** *Type:* String.
Example: “Code redirecting to exploit kits”.
A description of the matching code cluster.
- **id.** *Type:* String.
Example: “ek_redirector”.
The ID of the matching code cluster.
- **score.** *Type:* Integer.
Example: 70.
The maliciousness score associated to the matching cluster.

3.12.14 Applets Format

Information about applets.

The dictionary contains a (key, value) pair for every applet found during the analysis. The key is *file://* followed by the MD5 hash of the applet file. The value is a dictionary comprising the following fields:

- **contents.** *Type:* List of content details.
Details about individual applet contents.
 - **content_type.** *Type:* String.
Example: “application/x-unknown-mime-type”.
The content type of the applet element.
 - **length.** *Type:* Integer.
Example: 1024.
The length of the applet element.
 - **md5.** *Type:* Hexadecimal string.
Example: “e83bbd9d04cf15ee35a2911be221ae3b”.
The MD5 hash of the applet element.

- **name.** *Type:* String.
Example: “META-INF/MANIFEST.MF”.
The name of the applet element
- **sha1.** *Type:* Hexadecimal string.
Example: “6cf755e7ada47b9bea97dadd65f6140ed1863ca2”.
The SHA1 hash of the applet element.
- **result.** *Type:* Integer
Example: 1.
The applet class: 1 if malicious, 0 if benign.

3.13 Report Format *ll-static*

This analysis report format refers to a static analysis run of a PE or Mach-O executable file.

In addition to the report fields shared by all report formats (see *Analysis Report Format*) the report contains a number of different fields with details specific to the analysis run.

Reports may include fields not described here: they are to be considered as experimental or deprecated and SHOULD be ignored.

Report contents

- **analysis.** *Type:* Dictionary.
 - **file_information.** *Type:* Dictionary.
Basic information about the file contents.
 - * **md5.** *Type:* String.
md5 hash of analysis subject.
 - * **sha1.** *Type:* String.
sha1 hash of analysis subject.
 - * **sha256.** *Type:* String.
sha256 hash of the analysis subject.
 - * **size.** *Type:* Integer.
The analysis subject size (bytes).
 - * **ssdeep.** *Type:* String.
ssdeep fuzzy hash of the analysis subject. eg:
“1536:6UqqX4VONpYqNo+5DCGVM2/gXagwJm3rQcG/K:6UqqoVO/YqNf5DIVM2/gBwMrQf”
 - * **magic.** *Type:* String.
analysis subject magic description. eg: “Mach-O executable bundle”
 - **exif.** *Type:* Dictionary with Exiftool tag information; see *ExifTool EXE tag format* for details.
Exiftool EXE tag information.

- **authenticode.** *Type:* Dictionary.
Authenticode signature information for analysis subject.
- * **authentihash** *Type:* String.
Authentihash for analysis subject.
- **pefile.** *Type:* Dictionary.
Dictionary of information specific for PE files.
- * **exports.** *Type:* List of PE symbol exports; see *PE Export format* for details.
List of the symbols exported by the PE file.
- * **imports.** *Type:* List of symbol imports; see *PE Import format* for details.
List of the symbols imported by the PE file.
- * **file_version_properties.** *Type:* Dictionary; see *File Version Properties format* for details.
Information from the PE file version information resource
- * **header.** *Type:* Dictionary; see *PE Header format* for details.
PE header information.
- * **sections.** *Type:* List of sections in PE file; see *PE Section format* for details.
List of sections in PE file.
- * **resources.** *Type:* Dictionary; see *PE Resources format* for details.
Resources contained in PE file.
- * **debug_details.** *Type:* Dictionary; see *PE Debug Details format.* for details.
Debug information about PE file.
- * **imphash.** *Type:* String.
Import hash of PE file.

3.13.1 ExifTool EXE tag format

Dictionary with information on the EXE file from Exiftool. More information on these tags is available at <https://www.sno.phy.queensu.ca/~phil/exiftool/TagNames/EXE.html>. Not all analysis subjects will contain all tags.

- **file_type.** *Type:* String.
The type of the file analysis subject.
- **file_type_extension.** *Type:* String.
The File extension for the analysis subject.
- **mime_type.** *Type:* String.
MIME type for analysis subject.
- **machine_type.** *Type:* String.
CPU type for analysis subject.
- **timestamp.** *Type:* String.
File creation timestamp for analysis subject.

- **image_file_characteristics.** *Type:* String.
Bitwise characteristics flags for image file (hexadecimal).
- **pe_type.** *Type:* String.
Specific PE type.
- **linker_version.** *Type:* String.
Linker version.
- **code_size.** *Type:* Integer.
Size of source code.
- **initialized_data_size.** *Type:* Integer.
Size of initialized data.
- **uninitialized_data_size.** *Type:* Integer.
Size of uninitialized data.
- **entry_point.** *Type:* String.
Entrypoint (hexadecimal) address.
- **os_version.** *Type:* String.
OS Version.
- **image_version.** *Type:* String.
Image Version.
- **subsystem_version.** *Type:* String.
Subsystem Version.
- **subsystem.** *Type:* String.
Name of Subsystem.
- **file_version_number.** *Type:* String.
File Version.
- **product_version_number.** *Type:* String.
Product Version.
- **file_flags_mask.** *Type:* String.
Mask to apply to file flags (hexadecimal).
- **file_flags.** *Type:* String.
File Flags.
- **file_os.** *Type:* String.
Name of OS.
- **object_file_type.** *Type:* String.
Type of object file.
- **file_subtype.** *Type:* Integer.
Subtype of file.

- **build_date.** *Type:* String.
Date of build.
- **build_version.** *Type:* String.
Version of build.
- **character_set.** *Type:* String.
File character set.
- **comments.** *Type:* String.
Comment from PE resource string.
- **company_name.** *Type:* String.
Company name from PE resource string.
- **copyright.** *Type:* String.
Copyright message from PE resource string.
- **file_description.** *Type:* String.
File description from PE resource string.
- **file_version.** *Type:* String.
File version from PE resource string.
- **internal_name.** *Type:* String.
Internal name from PE resource string.
- **language_code.** *Type:* String.
Language code from PE resource string.
- **legal_copyright.** *Type:* String.
Legal copyright from PE resource string.
- **legal_trademarks.** *Type:* String.
Legal trademarks from PE resource string.
- **original_filename.** *Type:* String.
Original filename from PE resource string.
- **private_build.** *Type:* String.
Private build information from PE resource string.
- **product_name.** *Type:* String.
Product name from PE resource string.
- **product_version.** *Type:* String.
Product version from PE resource string.
- **special_build.** *Type:* String.
Special build info from PE resource string.
- **cpu_architecture.** *Type:* String.
CPU Architecture for MachO files.

- **cpu_byte_order.** *Type:* String.
CPU byte order for MachO files.
- **cpu_count.** *Type:* String.
CPU count for MachO files.
- **cpu_type.** *Type:* String.
CPU Type for MachO files (eg: 'x86').
- **cpu_sub_type.** *Type:* String.
CPU SubType for MachO files (eg: 'i386').
- **object_flags.** *Type:* String.
Object Flags for MachO files.

3.13.2 File Version Properties format

Dictionary of information from the PE file version information resource

- **copyright.** *Type:* String.
PE copyright information
- ***version.** *Type:* String.
PE version information
- **internal_name.** *Type:* String.
PE internal filename.
- **original_filename.** *Type:* String.
PE original filename.

3.13.3 PE Header format

Dictionary of PE header information.

- **compilation_timestamp.** *Type:* String.
Date/time of PE compilation.
- **number_of_sections.** *Type:* Integer.
Number of sections in PE file.
- **target_machine.** *Type:* String.
Target CPU type of PE file.
- **entry_point_address.** *Type:* String.
Entry point of PE file (hexadecimal).

3.13.4 PE Resources format

Dictionary of resources contained in PE file.

- **all** *Type*: List of resources; see *PE Resource format*. for details.
- **resource_count_by_language**. *Type*: List of resources by language; see *PE Resource By Language format*. for details.
- **resource_count_by_type**. *Type*: List of resources by resource type; see *PE Resource By Type format*. for details.

3.13.5 PE Import format

Dictionary of information about symbols imported by this PE file.

- **functions**. *Type*: List of imported functions; see *PE Function format* for details.

List of the functions imported by this PE file.

- **dll_name**. *Type*: String.
Name of the imported dll.

3.13.6 PE Function format

Dictionary of information about a function imported by this PE file.

- **name**. *Type*: String.
Name of the function.

3.13.7 PE Export format

Dictionary of information about symbols exported by this PE file.

- **ordinal**. *Type*: Integer.
PE symbol export ordinal index.
- **virtual_address**. *Type*: Integer.
The virtual address of the exported entry point.
- **name**. *Type*: String.
Name of the exported symbol.

3.13.8 PE Section format

Dictionary of information about the sections in the PE file.

- **name**. *Type*: String.
Name of the section.
- **virtual_address**: *Type*: String.
virtual address of this section (hexadecimal).

- **entropy:** *Type:* Floating-point number.
entropy of the section.
- **raw_size:** *Type:* String.
Actual size of the section (hexadecimal).
- **virtual_size** *Type:* String.
Virtual size of the section (hexadecimal).
- **md5.** *Type:* String.
md5 of the section.

3.13.9 PE Resource format.

Dictionary of information about a PE resource.

- **sha256.** *Type:* String.
sha256 hash of resource.
- **file_type.** *Type:* String.
File type of resource (eg: 'data' or 'ASCII text').
- **type.** *Type:* String.
Type of resource (eg: 'RT_ICON' or 'RT_MANIFEST').
- **language.** *Type:* String.
Language for resource.

3.13.10 PE Resource By Language format.

Dictionary of counts for resources by language.

- **count.** *Type:* Integer.
Number of resources in nominated language.
- **language.** *Type:* String.
Language of resource.

3.13.11 PE Resource By Type format.

Dictionary of counts for resources by resource type.

- **count.** *Type:* Integer.
Number of resources in nominated type.
- **type.** *Type:* String.
Type of resource (eg: 'RT_ICON' or 'RT_MANIFEST').

3.13.12 PE Debug Details format.

Debug information about PE file

- **pdb_path.** *Type:* String.
Path to PDB debug file.
- **guid.** *Type:* String.
GUID from PDB debug file.

3.14 Report Format *//ioc-json*

The Lastline Analyst API allows extracting *Indicators of Compromise* (IOC) from analysis reports. These IOC reports are available in three formats:

- Structured Threat Information Expression (STIX),
- Open Indicators of Compromise (OpenIOC), and
- Lastline IOC (format described in more detail below).

The OpenIOC and STIX formats are described in more detail in their respective documentation pages at <http://stixproject.github.io/releases/1.2/> for STIX and <http://schemas.mandiant.com> for OpenIOC v1.0 and v1.1 schemas.

IOC reports in Lastline IOC format are composed of two basic entities: *Matches* and *Rules*. A *Match* entity describes a single object (such as a file on the file system) with specific properties (such as the name of the file). A *Rule* combines multiple *Match* objects into complex expressions. Alternatively, a *Rule* can also combine one or more *Rule* objects. While some *Match* types describe objects that are available in STIX or OpenIOC, others do not have a direct mapping into the other IOC formats. The following table shows Lastline IOC *Match* types and their counterparts:

| Lastline | OpenIOC | STIX |
|-----------|--|----------------|
| Mutex | ProcessItem/HandleList/Handle/Name | MutexObj |
| Registry | RegistryItem/KeyPath, RegistryItem/Value | WinRegistryKey |
| File | FileItem/FullPath | WinFile |
| Autostart | None | None |

3.14.1 Report Structure

The Lastline IOC report contains a tree (dictionary) formatted using JSON or XML. This tree contains 2 main sections, containing a list of *Match* and *Rule* objects.

Matches

The *Matches* section contains a list of all basic entities that were extracted as *Indicator of Compromise*. That is, the analysis system found the behavior to be indicative/interesting in terms of the system modification.

Each entry is defined by a unique *name* attribute, as well as a *type*. Depending on the *type*, the entry has other, type-specific arguments (*args*) describing the entry in more detail.

Mutex

- *type* attribute value: *mutex_match*
- *args*:
 - *mutexname*: required string with global mutex name.

Registry

- *type* attribute value: *reg_match*
- *args*:
 - *reg_key*: required string with registry key path. Example: HKCU\Software\Microsoft\Windows.
 - *reg_value_name*: optional string with value name (note that this is not the data stored in the registry, but the name under which it is stored). Example: IsAutoLoadable.

File

- *type* attribute value: *file_system_match*
- *args*:
 - *filename*: required string with full path to file. Example: C:\mysample.txt
 - *md5*: optional string with md5 of file

Autostart

- *type* attribute value: *autostart*
- *args*:
 - *class*: one of *logon*, *explorer*, *bho*, *services*, *codecs*, *hijack*, *appinitdll*, *office*.
 - *filename*: required string with full path to file. Example: C:\mysample.txt
 - *md5*: optional string with md5 of file

Rules

The *Rules* section contains a list of boolean expressions combining individual *Match* or *Rule* entries into complex expressions. An expression contains one or more entries combined with boolean operators *and* or *or*.

A report may contain a special *Rule* named *root* to indicate one *Rule* entry that combines all other entries into one expression describing all objects described in this IOC report.

Examples

Example1

This example shows two *Match* entries, and each entry refers to a file on the file system using different parameters.

The objects are combined using the *OR* operator, meaning that the IOC should match if at least of the two files exists with the given properties:

```
{
  "matches": [
    {
      "name": "full_1",
      "type": "file_system_match",
      "args": {
        "filename": "c:\\windows\\systemself.exe"
      }
    }
  ]
}
```

```
    },
    {
      "name": "files_1",
      "type": "file_system_match",
      "args": {
        "md5": "6062fdc71440cb97db32c645627e181f",
        "filename": "c:\\windows\\systemself.exe"
      }
    }
  ],
  "rules": [
    {
      "name": "root",
      "match": "full_1 or files_1"
    }
  ]
}
```

Example2

This example shows two Microsoft Windows Registry elements. The first is described by its key, the other by its key and value name.

The objects are combined using the *AND* operator, meaning that the IOC should match if both registry entries exist with the given properties:

```
{
  "matches": [
    {
      "name": "full_1",
      "type": "reg_match",
      "args": {
        "reg_key": "HKLM\\test1"
      }
    },
    {
      "name": "full_2",
      "type": "reg_match",
      "args": {
        "reg_value_name": "ObjectId",
        "reg_key": "HKLM\\test2"
      }
    }
  ],
  "rules": [
    {
      "name": "all_indicators",
      "match": "full_1 and full_2"
    }
  ]
}
```

3.15 Report Format *ll-pcap*

This analysis report format refers to the analysis of a network traffic capture (pcap). The capture contents are analyzed to identify traffic that matches patterns of malicious connections (“signature detections”) and to identify connections

involving hosts that are known to our threat intelligence (“blacklist detections”).

In addition to the report fields shared by all report formats (see *Analysis Report Format*) the report contains a number of different fields with details specific to the analysis.

Reports may include fields not described here: they are to be considered as experimental or deprecated and SHOULD be ignored.

Report contents

- **domain_detections.** *Type:* List of detections on domains. See *Domain detections* for details.
A list providing information about detections that affected a domain.
- **ip_detections.** *Type:* List of detections on IP addresses. See *IP address detections* for details.
A list providing information about detections that affected an IP address.
- **url_detections.** *Type:* List of detections on URLs. See *URL detections* for details.
A list providing information about detections that affected a URL.

3.15.1 Detection information

Base information about a detection.

- **detection_type** *Type:* String.
Example: signature.
The type of detection. It can be either “signature”, indicating that the detection was caused by an IDS detection on the network traffic, or “blacklist”, indicating that the domain is known to be involved in malicious activity according to our threat intelligence.
- **threat_class** *Type:* String.
Example: drive-by.
The threat-class of this detection.
- **threat_name** *Type:* String.
Example: pseudo-darkleech redirection to exploit url.
The threat identified by this detection.
- **threat_severity** *Type:* Integer.
Example: 75.
Score between 0 and 100 indicating the severity of the detection.

3.15.2 Domain detections

A detection on a domain.

In addition to the generic detection information (see *Detection information*), it specifies:

- **domain** *Type:* String.
Example: example.com.
The domain involved in this detection.

3.15.3 IP address detections

A detection on an IP address.

In addition to the generic detection information (see *Detection information*), it specifies:

- **ip** *Type*: String.

Example: 93.184.216.34.

The IP address involved in this detection.

3.15.4 URL detections

A detection on a URL.

In addition to the generic detection information (see *Detection information*), it specifies:

- **url** *Type*: String.

Example: <http://example.com>.

The URL involved in this detection.

3.16 Report Format *ll-flash*

This analysis report format refers to a dynamic analysis run of a Flash file.

In addition to the report fields shared by all report formats (see *Analysis Report Format*) the report contains a number of different fields with details specific to the analysis run.

Reports may include fields not described here: they are to be considered as experimental or deprecated and SHOULD be ignored.

Report contents

- **callgraph**. *Type*: List of function call information; see *Callgraph Format*.

A list of ActionScript function calls that were observed during the analysis.

- **exploits**. *Type*: List of exploited vulnerabilities. See *Exploits Format* for details.

A list describing each vulnerability that was found to be exploited during the analysis.

- **generated_swfs**. *Type*: List of generated Flash files. See *Flash File Format* for details.

A list describing any Flash file that was dynamically generated during the analysis.

- **strings**. *Type*: List of strings. See *String Format* for details.

A list containing the strings observed during the analysis.

- **subject** *Type*: Dictionary describing the analysis subject. See *Flash File Format* for details.

3.16.1 Callgraph Format

A callgraph representing relationships between functions. The callgraph is recorded dynamically.

- **args**. *Type*: List of function arguments. See *Function Arguments and Return Value Format* for details.

The list of arguments that were passed to the current function.

- **callees.** *Type:* List of callees for the current function. See *Callgraph Format* for details.
The list of function calls called from the current function.
- **depth.** *Type:* Integer.
Example: 1
The depth in the callgraph.
- **name.** *Type:* String.
Example: “re52142333723350123423632234/re52142319223205123423632234”
The name of the function.
- **ret.** *Type:* Return value or null. See *Function Arguments and Return Value Format* for details.
Example: null
The return value of the function.
- **this.** *Type:* String.
Example: “0xfd20e80”
The address of the “this” object, in hexadecimal format.

3.16.2 Function Arguments and Return Value Format

A value passed as argument to a function or returned from a function.

- **typename.** *Type:* String.
Example: “int”
The type of the argument or return value.
- **value.** *Type:* String.
Example: “0x8”
The argument/return value.

3.16.3 Exploits Format

A vulnerability that was exploited during the analysis.

- **desc.** *Type:* String.
Example: “Buffer overflow in Flash Player via Blender data”
The vulnerability being exploited.
- **vendor** *Type:* String.
Example: “Adobe”
The vendor whose software contains the vulnerability.
- **vulnerability_id** *Type:* String.
Example: “CVE-2014-0515”
The public vulnerability ID, such as its CVE number.

- **vulnerability_url** *Type:* String.

Example: “https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0515”

A URL where more information about the vulnerability can be found.

3.16.4 Flash File Format

Information about a Flash file, either the original analysis subject or a Flash file that was dynamically generated during the analysis.

- **md5.** *Type:* hexadecimal string.

Example: 941f85f0ce9162a9b9531131b458c267

MD5 hash of the input file.

- **sha1.** *Type:* hexadecimal string.

Example: c511db6ae526e9ff2df60b2dba43dea1f8cdd591

SHA1 hash of the input file.

- **sha256.** *Type:* hexadecimal string.

Example: a820bb75a2d6fb069af2afc762ca6e30ab8c8b4d690ff880ed3a0a7b9bad36be

SHA256 hash of the input file.

- **compression.** *Type:* String.

Example: “zlib”

The compression type used by the input file.

- **filename.** *Type:* String.

Example: “941f85f0ce9162a9b9531131b458c267.swf”

The filename used during the submission.

- **frame_count.** *Type:* Integer.

Example: 1

The total number of frames in the Flash video.

- **num_tags.** *Type:* Integer.

Example: 12

The number of tags in the Flash file.

- **size.** *Type:* Integer.

Example: 29773

The number of bytes in the file.

- **tags.** *Type:* List of tags. See *Tag Format* for details.

The list of tags that compose the file.

- **version.** *Type:* Integer.

Example: 31

The Flash file version.

3.16.5 Strings Format

A string found during the Flash file execution.

- **value.** *Type:* String.

Example: “11,1,102,62”

The string value.

3.16.6 Tag Format

A tag in the Flash file. See the [Flash file format specification](#) for details.

- **name.** *Type:* String.

Example: “FileAttributes”

The name of the tag.

- **tagtype.** *Type:* int.

Example: 69

The tag ID.

Additional fields will be available, depending on the specific tag type.

3.17 Report Format *//doc*

This analysis report format refers to a static analysis run of a Microsoft Office document, Flash file or archive.

In addition to the report fields shared by all report formats (see [Analysis Report Format](#)) the report contains a number of different fields with details specific to the analysis run.

Reports may include fields not described here: they are to be considered as experimental or deprecated and SHOULD be ignored.

Report contents

- **analysis_subject: (optional).** *Type:* Dictionary.

- **document_name.** *Type:* String.

Name of the document that was analyzed.

- **file_size.** *Type:* Integer.

Size of the document (bytes).

- **md5.** *Type:* String.

md5 hash of the document.

- **sha1.** *Type:* String.

sha1 hash of the document.

- **analysis_metadata: (optional).** *Type:* List of analysis metadata; see [Document Metadata Format](#).

Document metadata extracted during analysis.

- **anomalies: (optional).** *Type:* List of dictionaries.

Anomalies detected in document script code.

- **description.** *Type:* String.

Example: “Evasion: VBA source code may have been altered”

Description of the detected anomaly.

- **location.** *Type:* String.

Example: “1f8fd8b6060284d5c47e26ec1021dd834af9b4655d289/Macros/VBA/NewMacros”

Location of the anomaly within the document.

- **macros: (optional).** *Type:* List of dictionaries.

Macros embedded in the document.

- **macro.** *Type:* String.

Macro code.

- **full_stream_path.** *Type:* String.

Location of the macro within the document.

- **streams: (optional).** *Type:* List of dictionaries.

Streams embedded in the document.

- **full_stream_path:** *Type:* String.

Location of the stream within the document.

- **child_streams:** *Type:* List of strings.

Locations of children of the stream within the document.

- **md5: (optional).** *Type:* Hexadecimal string.

A md5 hash of a file content.

- **sha1: (optional).** *Type:* Hexadecimal string.

A sha1 hash of a file content.

- **sha256: (optional).** *Type:* String.

sha256 hash of the analysis subject.

- **file_info: (optional).** *Type:* String.

Example: “MS Windows shortcut”.

A text description of stream type.

- **file_size: (optional).** *Type:* Integer.

A stream size in bytes.

- **mime: (optional).** *Type:* String.

MIME type of stream.

- **clsid: (optional).** *Type:* String.

Globally unique identifier for embedded (OLE) stream object.

3.17.1 Document Metadata Format

During the analysis run, the analysis engine extracts document metadata. This type extends the *Analysis Metadata Format* type.

Metadata contents

- **name: (optional).** *Type:* String.
Name given to the metadata.
- **filename: (optional).** *Type:* String.
Example: “desktop.ini”.
A file name. Could be absolute or relative path. Applies to metadata_type “extracted_file”.
- **abs_path: (optional).** *Type:* String.
Example: “C:\Users\desktop.ini”.
An absolute path the file. Applies to metadata_type “extracted_file”.

3.18 Report Descriptions

The type of analysis performed for artifacts submitted to the Analyst API depends on the type of the artifact (file or URL) as well as on the configuration of the Analyst API.

Each submission of a supported type is processed by one or more analysis engines. For example, a JavaScript file can be

- monitored by loading it in an instrumented browser,
- statically analyzed for anomalies in the script code, and
- run directly on the operating system via a script interpreter.

Each analysis engine performs its analysis, resulting in an analysis report each having a unique *report-UUID*, a *relevance* to the overall task classification, as well as a *description*.

The analysis report description provides details on what type of analysis was performed. Additionally it may contain optional information describing *where* the analysis was performed. In most cases this is the system to which the artifact was submitted - such as the Lastline hosted datacenter. For On-Premises customers that have the *cloud-analysis* component enabled, the report description may indicate that the same artifact was *executed on the Lastline cloud* and a relevant analysis report may be downloaded from the global Lastline analysis system. If this type of report is available and requested, the Analyst API transparently downloads the report from the Lastline backend and serves it like any other, locally-generated analysis report.

Note: If a report is available for download from the global Lastline (hosted) analysis system, it does *not* imply that the artifact (file or URL) was uploaded from the local On-Premises installation. Instead, it means that the artifact was already known to the global analysis system and a relevant analysis report was included as additional source of information for classification of the local analysis. For details, refer to the documentation of the *cloud-analysis* component in the [Lastline manuals](#).

CHILD TASKS

When analyzing a submitted artifact, the system may generate related sub- or child tasks as part of the analysis. Examples for such child tasks are files extracted from an archive, embedded programs extracted from a document, files found during the analysis of a URL, network traffic generated during the analysis or customized run of a document analysis.

While these child tasks are standalone analysis runs, the analysis outcome/classification of the child task may also influence the classification of the originally submitted artifact. For example, if a file is extracted from an archive and this file is found to be malicious, the archive is also considered to be malicious.

The analysis results of these child tasks are linked to the original analysis and accessible as part of the *child_task* field of the analysis result.

Child task contents

- **task_uuid.** *Type:* Hexadecimal string.

Example: 7065a3ba0c729ad5981a1e1072df710d.

Unique identifier for this analysis submission.

- **score.** *Type:* Integer.

Example: 75.

Score between 0 and 100 indicating maliciousness of the observed behavior (0=benign, 100=malicious). See **Contents of response** for details.

- **tag.** *Type:* String.

Example: “file extracted from archive”.

A short description of the child task’s type.

- **parent_report_uuid: (optional).** *Type:* String.

Example: c6600954ea584d0a8912e4be80609122

If the child task is linked to a specific analysis report, such as the analysis inside the Windows 7 sandbox, of the (original/parent) task, the *parent_report_uuid* points to the corresponding report.

SAMPLE API CLIENTS

To make using the Lastline Analyst API easier, we provide client code in Python. Specifically, we provide the following:

- **Analyst API client:** an API client providing detailed documentation for each function available in the API, and serving as basis for using the Lastline Analyst API in Python.
- **Analysis Client Shell:** an easy way to explore the Lastline Analyst API from an interactive Python shell.
- **analyze_artifacts:** two simple command-line programs that use the API client to analyze URLs and files found (recursively) in a directory. This is also available as a stand-alone windows executable.
- **Application Bundle Module:** Python utility code for building *application bundles*, which allow detailed customization of the analysis.

5.1 Analyst API client

This is a Python client for the Lastline Analyst API.

The `AnalysisClient` class implements the client side of the Lastline Analyst API methods. It can be imported into Python client code that uses the API.

The client is available at https://analysis.lastline.com/docs/llapi_client/analysis_apiclient.py .

5.1.1 Requirements

The Analysis API client requires:

- Python 2.7.
- The python requests module (tested with version 2.2.1).
- The python simplejson module (tested with version 3.6.5).
- To use the client as a python shell, the ipython module (tested with version 2.4.1).

Required python modules can be installed using tools such as apt, pip, or easy_install, e.g.:

```
apt-get install python-pycurl=7.19.0-4ubuntu3
pip install ipython==2.4.1
easy_install requests==2.2.1
```

Note: You may want to consider installing the API client and its dependencies inside an isolated environment, such as a container, schroot, or VirtualEnv. This allows experimenting with the Lastline APIs without affecting system libraries/modules.

5.1.2 Changelog

The changelog only reflects backwards-incompatible changes; new functionality may not be reflected in all cases

- **2016-10-05: Stop download of full report details during submission** Submission functions, such as `submit_file()`, `submit_file_hash()`, or `submit_url()`, now default to `full_report_score=ANALYSIS_API_NO_REPORT_DETAILS` (constant for -1), which disables automatic download of the full, detailed analysis report if a cached result is immediately available. To access the full analysis report, use `get_result()` with the `task_uuid` returned as part of the submission result.
- **2016-10-28: Move API client shell to dedicated script.** The API client shell is now available via `analysis_apiclient_shell.py`, which provides easier access to helper modules provided by the API client module.

5.1.3 Analysis Client Shell

In addition to the client, an API shell allows running the client from the command line. This provides an interactive shell for manually sending requests to the Lastline Analyst API, and it can be used to experiment with the API for analyzing files or URLs. For details, refer to the *API Client Shell documentation*.

5.1.4 Analyst API Client Classes

```
class llapi_client.analysis_apiclient.AnalysisClientBase (base_url, use_cdn=None,  
logger=None, config=None)
```

A client for the Lastline analysis API.

This is an abstract base class: concrete subclasses just need to implement the `_api_request` method to actually send the API request to the server.

Parameters

- **base_url** – URL where the lastline analysis API is located. (required)
- **logger** – if provided, should be a python logging.Logger object or object with similar interface.

```
submit_file (file_stream, download_ip=None, download_port=None, download_url=None,  
download_host=None, download_path=None, download_agent=None, down-  
load_referer=None, download_request=None, full_report_score=-1, by-  
pass_cache=None, delete_after_analysis=None, backend=None, analy-  
sis_timeout=None, analysis_env=None, allow_network_traffic=None, filename=None,  
keep_file_dumps=None, keep_memory_dumps=None, keep_behavior_log=None,  
push_to_portal_account=None, raw=False, verify=True, server_ip=None,  
server_port=None, server_host=None, client_ip=None, client_port=None,  
is_download=True, protocol='http', apk_package_name=None, password=None,  
password_candidates=None, report_version=None, analysis_task_uuid=None, anal-  
ysis_engine=None, task_metadata=None, priority=None, bypass_prefilter=None,  
fast_analysis=None)
```


Submit a file by uploading it.

For return values and error codes please see `malscape_service.api.views.analysis.submit_file()`.

Parameters

- **file_stream** – file-like object containing the file to upload.
- **download_ip** – DEPRECATED! Use `server_ip` instead.
- **download_port** – DEPRECATED! Use `server_port` instead.
- **download_url** – DEPRECATED! replaced by the `download_host` and `download_path` parameters
- **download_host** – hostname of the server-side endpoint of the connection, as a string of bytes (not unicode).
- **download_path** – host path from which the submitted file was originally downloaded, as a string of bytes (not unicode)
- **download_agent** – HTTP user-agent header that was used when the submitted file was originally downloaded, as a string of bytes (not unicode)
- **download_referer** – HTTP referer header that was used when the submitted file was originally downloaded, as a string of bytes (not unicode)
- **download_request** – full HTTP request with which the submitted file was originally downloaded, as a string of bytes (not unicode)
- **full_report_score** – if set, this value (between -1 and 101) determines starting at which scores a full report is returned. -1 and 101 indicate “never return full report”; 0 indicates “return full report at all times”
- **bypass_cache** – if True, the API will not serve a cached result. NOTE: This requires special privileges.
- **delete_after_analysis** – if True, the backend will delete the file after analysis is done (and noone previously submitted this file with this flag set)
- **analysis_timeout** – timeout in seconds after which to terminate analysis. The analysis engine might decide to extend this timeout if necessary. If all analysis subjects terminate before this timeout analysis might be shorter
- **analysis_env** – environment in which to run analysis. This includes the operating system as well as version of tools such as Microsoft Office. Example usage: - windows7:office2003, or - windowsxp By default, analysis will run on all available operating systems using the most applicable tools.
- **allow_network_traffic** – if False, all network connections will be redirected to a honeypot. Requires special permissions.
- **filename** – filename to use during analysis. If none is passed, the analysis engine will pick an appropriate name automatically. An easy way to pass this value is to use `'file_stream.name'` for most file-like objects
- **keep_file_dumps** – if True, all files generated during analysis will be kept for post-processing. NOTE: This can generate large volumes of data and is not recommended. Requires special permissions
- **keep_memory_dumps** – if True, all buffers allocated during analysis will be kept for post-processing. NOTE: This can generate large volumes of data and is not recommended. Requires special permissions

- **keep_behavior_log** – if True, the raw behavior log extracted during analysis will be kept for post-processing. NOTE: This can generate *very very* large volumes of data and is not recommended. Requires special permissions
- **push_to_portal_account** – if set, a successful submission will be pushed to the web-portal using the specified username
- **backend** – DEPRECATED! Don't use
- **verify** – if False, disable SSL-certificate verification
- **raw** – if True, return the raw JSON results of the API query
- **server_ip** – ASCII dotted-quad representation of the IP address of the server-side endpoint.
- **server_port** – integer representation of the port number of the server-side endpoint of the flow tuple.
- **server_host** – DEPRECATED! Don't use
- **client_ip** – ASCII dotted-quad representation of the IP address of the client-side endpoint.
- **client_port** – integer representation of the port number of the client-side endpoint of the flow tuple.
- **is_download** – Boolean; True if the transfer happened in the server -> client direction, False otherwise (client -> server).
- **protocol** – app-layer protocol in which the file got transferred. Short ASCII string.
- **report_version** – Version name of the Report that will be returned (optional);
- **apk_package_name** – package name for APK files. Don't specify manually.
- **password** – password used to analyze password-protected or encrypted content (such as archives or documents)
- **password_candidates** – List of passwords used to analyze password-protected or encrypted content (such as archives or documents)
- **analysis_task_uuid** – if the call is used to create a child task, it specifies the current analysis task UUID; None otherwise. Lastline-internal/do not use.
- **analysis_engine** – if `analysis_task_uuid` is provided, it specifies the sandbox it refers to; None otherwise. Lastline-internal/do not use.
- **task_metadata** – optional task-metadata to upload. Requires special permissions; Lastline-internal/do not use
- **priority** – Priority level to set for this analysis. Priority should be between 1 and 10 (1 is the lowest priority, 10 is the highest) Setting priority to any value other than 1 requires special permissions.
- **bypass_prefilter** – Boolean; If True, file is submitted to all supported analysis components without prior static analysis. Requires special permissions.
- **fast_analysis** – Boolean; If True, file is submitted only to fast analyzers (static)

Raises

- **[AnalysisAPIError](#)** – Analysis API returns HTTP error or error code (and 'raw' not set)
- **[CommunicationError](#)** – Error contacting Lastline Analyst API.

submit_file_hash (*md5=None, sha1=None, sha256=None, download_ip=None, download_port=None, download_url=None, download_host=None, download_path=None, download_agent=None, download_referer=None, download_request=None, full_report_score=-1, bypass_cache=None, password=None, password_candidates=None, backend=None, require_file_analysis=True, mime_type=None, analysis_timeout=None, analysis_env=None, allow_network_traffic=None, filename=None, keep_file_dumps=None, keep_memory_dumps=None, keep_behavior_log=None, push_to_portal_account=None, raw=False, verify=True, server_ip=None, server_port=None, server_host=None, client_ip=None, client_port=None, is_download=True, protocol='http', apk_package_name=None, report_version=None, analysis_task_uid=None, analysis_engine=None, task_metadata=None, priority=None, bypass_prefilter=None, fast_analysis=None*)

Submit a file by hash.

One of the md5, sha1, or sha256 parameters must be provided. If both are provided, they should be consistent.

For return values and error codes please see [malscape_service.api.views.analysis.submit_file\(\)](#).

Parameters

- **md5** – md5 hash of file.
- **sha1** – sha1 hash of file.
- **sha256** – sha256 hash of file.
- **download_ip** – DEPRECATED! Use `server_ip` instead.
- **download_port** – DEPRECATED! Use `server_port` instead.
- **download_url** – DEPRECATED! replaced by the `download_host` and `download_path` parameters
- **download_host** – hostname of the server-side endpoint of the connection, as a string of bytes (not unicode).
- **download_path** – host path from which the submitted file was originally downloaded, as a string of bytes (not unicode)
- **download_agent** – HTTP user-agent header that was used when the submitted file was originally downloaded, as a string of bytes (not unicode)
- **download_referer** – HTTP referer header that was used when the submitted file was originally downloaded, as a string of bytes (not unicode)
- **download_request** – full HTTP request with which the submitted file was originally downloaded, as a string of bytes (not unicode)
- **full_report_score** – if set, this value (between -1 and 101) determines starting at which scores a full report is returned. -1 and 101 indicate “never return full report”; 0 indicates “return full report at all times”
- **bypass_cache** – if True, the API will not serve a cached result. NOTE: This requires special privileges.
- **password** – password used to analyze password-protected or encrypted content (such as archives or documents)

- **password_candidates** – List of passwords used to analyze password-protected or encrypted content (such as archives or documents)
- **require_file_analysis** – if True, the submission requires an analysis run to be started. If False, the API will attempt to base a decision solely on static information such as download source reputation and hash lookups. Requires special permissions; Lastline-internal/do not use
- **mime_type** – the mime-type of the file; This value should be set when `require_file_analysis` is True to enforce getting the most information available
- **analysis_timeout** – timeout in seconds after which to terminate analysis. The analysis engine might decide to extend this timeout if necessary. If all analysis subjects terminate before this timeout analysis might be shorter
- **analysis_env** – environment in which to run analysis. This includes the operating system as well as version of tools such as Microsoft Office. Example usage: - windows7:office2003, or - windowsxp By default, analysis will run on all available operating systems using the most applicable tools.
- **allow_network_traffic** – if False, all network connections will be redirected to a honeypot. Requires special permissions.
- **filename** – filename to use during analysis. If none is passed, the analysis engine will pick an appropriate name automatically. An easy way to pass this value is to use `'file_stream.name'` for most file-like objects
- **keep_file_dumps** – if True, all files generated during analysis will be kept for post-processing. NOTE: This can generate large volumes of data and is not recommended. Requires special permissions
- **keep_memory_dumps** – if True, all buffers allocated during analysis will be kept for post-processing. NOTE: This can generate *very* large volumes of data and is not recommended. Requires special permissions
- **keep_behavior_log** – if True, the raw behavior log extracted during analysis will be kept for post-processing. NOTE: This can generate *very very* large volumes of data and is not recommended. Requires special permissions
- **push_to_portal_account** – if set, a successful submission will be pushed to the web-portal using the specified account
- **backend** – DEPRECATED! Don't use
- **verify** – if False, disable SSL-certificate verification
- **raw** – if True, return the raw json results of the API query
- **server_ip** – ASCII dotted-quad representation of the IP address of the server-side endpoint.
- **server_port** – integer representation of the port number of the server-side endpoint of the flow tuple.
- **server_host** – DEPRECATED! Don't use
- **client_ip** – ASCII dotted-quad representation of the IP address of the client-side endpoint.
- **client_port** – integer representation of the port number of the client-side endpoint of the flow tuple.

- **is_download** – Boolean; True if the transfer happened in the server -> client direction, False otherwise (client -> server).
- **protocol** – app-layer protocol in which the file got transferred. Short ASCII string.
- **apk_package_name** – package name for APK files. Don't specify manually.
- **report_version** – Version name of the Report that will be returned (optional);
- **analysis_task_uuid** – if the call is used to create a child task, it specifies the current analysis task UUID; None otherwise. Lastline-internal/do not use.
- **analysis_engine** – if **analysis_task_uuid** is provided, it specifies the sandbox it refers to; None otherwise. Lastline-internal/do not use.
- **task_metadata** – optional task-metadata to upload. Requires special permissions; Lastline-internal/do not use
- **priority** – Priority level to set for this analysis. Priority should be between 1 and 10 (1 is the lowest priority, 10 is the highest). Setting priority to any value other than 1 requires special permissions.
- **bypass_prefilter** – Boolean; If True, file is submitted to all supported analysis components without prior static analysis. Requires special permissions.
- **fast_analysis** – Boolean; If True, file is submitted only to fast analyzers (static)

Raises

- **AnalysisAPIError** – Analysis API returns HTTP error or error code (and 'raw' not set)
- **CommunicationError** – Error contacting Lastline Analyst API.

submit_url (*url*, *referer=None*, *full_report_score=-1*, *bypass_cache=None*, *backend=None*, *analysis_timeout=None*, *push_to_portal_account=None*, *raw=False*, *verify=True*, *user_agent=None*, *report_version=None*, *analysis_task_uuid=None*, *analysis_engine=None*, *priority=None*, *task_metadata=None*, *fast_analysis=None*, *password_candidates=None*)

Submit a url.

For return values and error codes please see `malscape_service.api.views.analysis.submit_url()`.

Parameters

- **url** – url to analyze
- **referer** – referer header to use for analysis
- **full_report_score** – if set, this value (between -1 and 101) determines starting at which scores a full report is returned. -1 and 101 indicate “never return full report”; 0 indicates “return full report at all times”
- **bypass_cache** – if True, the API will not serve a cached result. NOTE: This requires special privileges.
- **analysis_timeout** – timeout in seconds after which to terminate analysis. The analysis engine might decide to extend this timeout if necessary. If all analysis subjects terminate before this timeout analysis might be shorter
- **push_to_portal_account** – if set, a successful submission will be pushed to the web-portal using the specified account
- **backend** – DEPRECATED! Don't use

- **verify** – if False, disable SSL-certificate verification
- **raw** – if True, return the raw JSON results of the API query
- **report_version** – Version name of the Report that will be returned (optional);
- **user_agent** – user agent header to use for analysis
- **analysis_task_uuid** – if the call is used to create a child task, it specifies the current analysis task UUID; None otherwise. Lastline-internal/do not use.
- **analysis_engine** – if **analysis_task_uuid** is provided, it specifies the sandbox it refers to; None otherwise. Lastline-internal/do not use.
- **priority** – Priority level to set for this analysis. Priority should be between 1 and 10 (1 is the lowest priority, 10 is the highest). Setting priority to any value other than 1 requires special permissions.
- **task_metadata** – optional task-metadata to upload. Requires special permissions; Lastline-internal/do not use
- **fast_analysis** – Boolean; If True, url is submitted only to fast analyzers (static)
- **password_candidates** – List of passwords used to analyze password-protected or encrypted content from the URL.

Raises

- **AnalysisAPIError** – Analysis API returns HTTP error or error code (and ‘raw’ not set)
- **CommunicationError** – Error contacting Lastline Analyst API.

get_result (*uuid*, *report_uuid=None*, *full_report_score=None*, *include_scoring_components=None*, *raw=False*, *requested_format='json'*, *verify=True*, *report_version=None*, *allow_datacenter_redirect=None*)

Get results for a previously submitted analysis task.

For return values and error codes please see `mal scape_service.api.views.analysis.get_results()`.

Parameters

- **uuid** – the unique identifier of the submitted task, as returned in the `task_uuid` field of submit methods.
- **report_uuid** – if set, include this report in the result.
- **full_report_score** – if set, this value (between -1 and 101) determines starting at which scores a full report is returned. -1 and 101 indicate “never return full report”; 0 indicates “return full report at all times”
- **include_scoring_components** – if True, the result will contain details of all components contributing to the overall score. Requires special permissions
- **raw** – if True, return the raw JSON/XML results of the API query.
- **requested_format** – JSON, XML, PDF, or RTF. If format is not JSON, this implies *raw*.
- **report_version** – Version of the report to be returned If *report_uuid* is not specified, this parameter is ignored. (optional)
- **allow_datacenter_redirect** – If False, redirection to other datacenters prevented.

Raises

- ***AnalysisAPIError*** – Analysis API returns HTTP error or error code (and ‘raw’ not set)
- ***CommunicationError*** – Error contacting Lastline Analyst API.

get_result_summary (*uuid*, *raw=False*, *requested_format='json'*, *score_only=False*, *verify=True*, *allow_datacenter_redirect=None*)
Get result summary for a previously submitted analysis task.

For return values and error codes please see *malscape_service.api.views.analysis.get_result()*.

Parameters

- **uuid** – the unique identifier of the submitted task, as returned in the *task_uuid* field of submit methods.
- **raw** – if True, return the raw JSON/XML results of the API query.
- **requested_format** – JSON or XML. If format is not JSON, this implies *raw*.
- **score_only** – if True, return even less data (only score and threat/threat-class classification).
- **allow_datacenter_redirect** – If False, redirection to other datacenters prevented.

Raises

- ***AnalysisAPIError*** – Analysis API returns HTTP error or error code (and ‘raw’ not set)
- ***CommunicationError*** – Error contacting Lastline Analyst API.

get_result_artifact (*uuid*, *report_uuid*, *artifact_name*, *password_protected=None*, *raw=False*, *verify=True*, *allow_datacenter_redirect=None*)
Get artifact generated by an analysis result for a previously submitted analysis task.

NOTE: Consider using *get_report_artifact()* if the artifact is bound to a specific analysis report (which it is in practically all cases).

Parameters

- **uuid** – the unique identifier of the submitted task, as returned in the *task_uuid* field of submit methods.
- **report_uuid** – the unique report identifier returned as part of the dictionary returned by *get_result()*.
- **artifact_name** – the name of the artifact as mentioned in the given report in the dictionary returned by *get_result()*.
- **password_protected** (*str*) – If provided, use this password to create a zip which will contain the artifact being fetched. The password provided should be using only ASCII characters and have max length of 128 characters
- **raw** – if True, return the raw JSON/XML results of the API query.
- **allow_datacenter_redirect** – If False, redirection to other datacenters prevented.

Raises

- **AnalysisAPIError** – Analysis API returns HTTP error or error code (and ‘raw’ not set)
- **CommunicationError** – Error contacting Lastline Analyst API.

get_report_artifact (*uuid, report_uuid, artifact_name, password_protected=None, verify=True, allow_datacenter_redirect=None*)

Get artifact generated by an analysis result for a previously submitted analysis task.

Parameters

- **uuid** (*str*) – the unique identifier of the submitted task, as returned in the `task_uuid` field of submit methods.
- **report_uuid** (*str*) – the unique report identifier returned as part of the dictionary returned by `get_result()`.
- **artifact_name** (*str*) – the name of the artifact as mentioned in the given report in the dictionary returned by `get_result()`.
- **password_protected** (*str*) – If provided, use this password to create a zip which will contain the artifact being fetched. The password provided should be using only ASCII characters and have max length of 128 characters
- **allow_datacenter_redirect** – If False, redirection to other datacenters prevented.

Returns A stream containing the artifact content

Return type stream

Raises

- **AnalysisAPIError** – Analysis API returns HTTP error or error code (and ‘raw’ not set)
- **CommunicationError** – Error contacting Lastline Analyst API.

get_completed (*after, before=None, raw=False, verify=True, include_score=False*)

Get the list of uuids of tasks that were completed within a given time frame.

The main use-case for this method is to periodically request a list of uuids completed since the last time this method was invoked, and then fetch each result with `get_result()`.

Date parameters to this method can be:

- date string: `%Y-%m-%d`
- datetime string: `‘%Y-%m-%d %H:%M:%S’`
- `datetime.datetime` object

All times are in UTC.

For return values and error codes please see `mal scape_service.api.views.analysis.get_completed()`.

Parameters

- **after** – Request tasks completed after this time.
- **before** – Request tasks completed before this time.
- **include_score** – If True, the response contains scores together with the task-UUIDs that have completed
- **raw** – if True, return the raw JSON results of the API query.

Raises

- **AnalysisAPIError** – Analysis API returns HTTP error or error code (and ‘raw’ not set)
- **CommunicationError** – Error contacting Lastline Analyst API.

get_completed_with_metadata (*after, before=None, raw=False, verify=True*)

Get the list of dictionaries, each containing a uuid for a task that was completed within a given time frame, the resulting score, and additional task_metadata

The main use-case for this method is to periodically request a list of dictionaries containing information about each task, such as the score and task_metadata. Then, additional information can be retrieved for a task with *get_result()*

Date parameters to this method can be:

- date string: ‘%Y-%m-%d’
- datetime string: ‘%Y-%m-%d %H:%M:%S’
- datetime.datetime object

All times are in UTC.

For return values and error codes please see *malscape_service.api.views.analysis.get_completed_with_metadata()*.

Parameters

- **after** – Request tasks completed after this time.
- **before** – Request tasks completed before this time.
- **raw** – if True, return the raw JSON results of the API query.

Raises

- **AnalysisAPIError** – Analysis API returns HTTP error or error code (and ‘raw’ not set)
- **CommunicationError** – Error contacting Lastline Analyst API.

get_progress (*uuid, raw=False, allow_datacenter_redirect=None*)

Get a progress estimate for a previously submitted analysis task.

For return values and error codes please see *malscape_service.api.views.analysis.get_results()*.

Parameters

- **uuid** – the unique identifier of the submitted task, as returned in the task_uuid field of submit methods.
- **raw** – if True, return the raw JSON/XML results of the API query.
- **requested_format** – JSON or XML. If format is not JSON, this implies *raw*.
- **allow_datacenter_redirect** – If False, redirection to other datacenters prevented.

Raises

- **AnalysisAPIError** – Analysis API returns HTTP error or error code (and ‘raw’ not set)
- **CommunicationError** – Error contacting Lastline Analyst API.

is_risky_analysis_artifact (*report_uuid, artifact_name, task_uuid=None, raw=False, verify=True, allow_datacenter_redirect=None*)

Check if the artifact can potentially be malicious using the artifact information.

Parameters

- **report_uuid** (*str*) – Identifier of the requested report to which the artifact is assigned
- **artifact_name** (*str*) – Identifier of task artifact
- **task_uuid** (*str/None*) – Unique identifier for the task that analyzed the artifact. If not present, will only look for artifact in local datacenter.
- **raw** (*bool*) – if True, return the raw JSON results of the API query.
- **verify** (*bool*) – if True, verify ssl, otherwise False
- **allow_datacenter_redirect** (*bool/None*) – If False, redirection to other datacenters prevented.

Returns True if the artifact is risky, False otherwise

Return type bool

Raises

- **AnalysisAPIError** – Analysis API returns HTTP error or error code (and ‘raw’ not set)
- **CommunicationError** – Error contacting Lastline Analyst API.
- **InvalidArtifactError** – Invalid artifact uuid.

```
class llapi_client.analysis_apiclient.AnalysisClient (base_url, key, api_token, logger=None, ca_bundle=None, verify_ssl=True, use_curl=False, timeout=60, use_cdn=None, proxies=None, config=None)
```

Client for the Analysis API.

A client for the Analysis API that accesses the API through the web, using key and api token for authentication, and the python requests module for sending requests.

NOTE: This class is not thread safe

```
class llapi_client.analysis_apiclient.SubmissionHelper (analysis_client, logger=None, num_retries=10)
```

Helper class for handling submission and task retrieval

submit_file_stream (*file_stream, **kwargs*)

Submit a file for analysis and retrieve results if they are immediately available. Additional parameters passed to this function are forwarded to the client (see *submit_file_hash* or *submit_file*).

NOTE: To avoid a race-condition between submission and polling for results, use the following approach:

```
helper = SubmissionHelper(<client>)
ts = helper.get_api_utc_timestamp()
submission = helper.submit_file_stream(<stream>)
helper.wait_for_completion_of_submission(submission, ts)
```

or use the *submit_file_streams_and_wait_for_completion()* helper function.

NOTE: You may provide any of the parameters - `file_md5`, - `file_sha1`, or - `file_sha256` to avoid repeated file-hash calculations. Any hash not provided will be generated from the given file-stream.

Parameters `file_stream` (*stream*) – Stream to submit

Returns Submission results

Return type *SubmittedFileTask*

Raises

- ***AnalysisAPIError*** – Analysis API returns HTTP error or error code (and ‘raw’ not set)
- ***CommunicationError*** – Error contacting Lastline Analyst API.

submit_filename (*filename*, ****kwargs**)

Submit a file for analysis and retrieve results if they are immediately available. Additional parameters passed to this function are forwarded to the client (see *submit_file_hash* or *submit_file*).

NOTE: To avoid a race-condition between submission and polling for results, use the following approach:

```
helper = SubmissionHelper(<client>)
ts = helper.get_api_utc_timestamp()
submission = helper.submit_filename(<filename>)
helper.wait_for_completion_of_submission(submission, ts)
```

or use the *submit_filenames_and_wait_for_completion()* helper function.

Parameters `filename` (*str*) – File on the local filesystem to submit

Returns Submission results

Return type *SubmittedFileTask*

Raises

- ***AnalysisAPIError*** – Analysis API returns HTTP error or error code (and ‘raw’ not set)
- ***CommunicationError*** – Error contacting Lastline Analyst API.

submit_url (*url*, ****kwargs**)

Submit a URL for analysis and retrieve results if they are immediately available. Additional parameters passed to this function are forwarded to the client (see *submit_url*).

NOTE: To avoid a race-condition between submission and polling for results, use the following approach:

```
helper = SubmissionHelper(<client>)
ts = helper.get_api_utc_timestamp()
submission = helper.submit_url(<url>, referer=<referer>)
helper.wait_for_completion_of_submission(submission, ts)
```

or use the *submit_urls_and_wait_for_completion()* helper function.

Parameters `url` (*str*) – URL to submit

Returns Submission results

Return type *SubmittedURLTask*

Raises

- ***AnalysisAPIError*** – Analysis API returns HTTP error or error code (and ‘raw’ not set)

- **CommunicationError** – Error contacting Lastline Analyst API.

wait_for_completion_of_submission (*submission*, *start_timestamp*,
wait_completion_interval_seconds=15,
wait_completion_max_seconds=None, verify=True)

Wait for completion of a given tasks.

Parameters

- **submission** (*SubmittedTask*) – A submitted task. This object is updated in place with result data
- **start_timestamp** (*datetime.datetime*) – UTC timestamp before the first submission has happened. Use *self.get_api_utc_timestamp()* to retrieve or use the *submission_timestamp* returned from the submission.
- **wait_completion_interval_seconds** (*float*) – How long to wait between polls for completion
- **wait_completion_max_seconds** (*float*) – Don't wait for longer than this many seconds for completion. If *None* is specified, wait forever
- **verify** (*bool*) – if *False*, disable SSL-certificate verification

Raises

- **WaitResultTimeout** – Waiting for results timed out
- **AnalysisAPIError** – Analysis API returns HTTP error or error code (and 'raw' not set)
- **CommunicationError** – Error contacting Lastline Analyst API.

submit_file_streams_and_wait_for_completion (*file_streams*,
wait_completion_interval_seconds=15,
wait_completion_max_seconds=None,
***kwargs*)

Submit a list of files and wait for completion: For each file, submit the file for analysis, wait for completion, and retrieve results. Additional parameters passed to this function are forwarded to the client (see *submit_file_hash* or *submit_file*).

Parameters

- **file_streams** (*list('stream')*) – List of streams to submit
- **wait_completion_interval_seconds** (*float*) – How long to wait between polls for completion
- **wait_completion_max_seconds** (*float*) – Don't wait for longer than this many seconds for completion. If *None* is specified, wait forever. NOTE: If waiting times out, the result will contain elements whose score is set to *None*. This method does *not* raise *WaitResultTimeout* to allow retrieving the result even when waiting for completion timed out.

Returns Dictionary of results

Return type *dict('SubmittedFileTask')*

Raises

- **AnalysisAPIError** – Analysis API returns HTTP error or error code (and 'raw' not set)
- **CommunicationError** – Error contacting Lastline Analyst API.

submit_filenames_and_wait_for_completion (*filenames*, *wait_completion_interval_seconds=15*,
wait_completion_max_seconds=None,
***kwargs*)

Submit a list of files and wait for completion: For each file, submit the file for analysis, wait for completion, and retrieve results. Additional parameters passed to this function are forwarded to the client (see *submit_file_hash* or *submit_file*).

Parameters

- **filenames** (*list('str')*) – List of files on the local filesystem to submit
- **wait_completion_interval_seconds** (*float*) – How long to wait between polls for completion
- **wait_completion_max_seconds** (*float*) – Don't wait for longer than this many seconds for completion. If None is specified, wait forever. NOTE: If waiting times out, the result will contain elements whose score is set to *None*. This method does *not* raise *WaitResultTimeout* to allow retrieving the result even when waiting for completion timed out.

Returns Dictionary of results

Return type *dict('SubmittedFileTask')*

Raises

- **AnalysisAPIError** – Analysis API returns HTTP error or error code (and 'raw' not set)
- **CommunicationError** – Error contacting Lastline Analyst API.

submit_urls_and_wait_for_completion (*urls*, *wait_completion_interval_seconds=15*,
wait_completion_max_seconds=None, ***kwargs*)

Submit a list of URLs and wait for completion: For each URL, submit the URL for analysis, wait for completion, and retrieve results. Additional parameters passed to this function are forwarded to the client (see *submit_url*).

Parameters

- **urls** (*list('str')*) – List of URLs to submit
- **wait_completion_interval_seconds** (*float*) – How long to wait between polls for completion
- **wait_completion_max_seconds** (*float*) – Don't wait for longer than this many seconds for completion. If None is specified, wait forever

Returns Dictionary of results

Return type *dict('SubmittedURLTask')*

Raises

- **AnalysisAPIError** – Analysis API returns HTTP error or error code (and 'raw' not set)
- **CommunicationError** – Error contacting Lastline Analyst API.

wait_for_completion (*submissions*, *start_timestamp*, *wait_completion_interval_seconds=15*,
wait_completion_max_seconds=None, *verify=True*)

Wait for completion of a given dictionary of tasks.

NOTE: Results are filled in in provided *submissions* dictionary.

Parameters

- **submissions** (*dict*(*id*: *SubmittedTask*)) – Dictionary of submissions: submission identifier to *SubmittedTask* mapping. NOTE: The submission identifier can be an arbitrary value unique to the dictionary
- **start_timestamp** (*datetime.datetime*) – UTC timestamp before the first submission has happened. Use *self.get_api_utc_timestamp()* to retrieve or use the *submission_timestamp* returned from the submission.
- **wait_completion_interval_seconds** (*float*) – How long to wait between polls for completion
- **wait_completion_max_seconds** (*float*) – Don't wait for longer than this many seconds for completion. If *None* is specified, wait forever
- **verify** (*bool*) – if *False*, disable SSL-certificate verification

Raises

- **WaitResultTimeout** – Waiting for results timed out
- **AnalysisAPIError** – Analysis API returns HTTP error or error code (and 'raw' not set)
- **CommunicationError** – Error contacting Lastline Analyst API.

5.1.5 Exceptions

class `llapi_client.analysis_apiclient.AnalysisAPIError` (*msg*, *error_code*)
Analysis API returned an error.

The *error_code* member of this exception is the *error code returned by the API*.

class `llapi_client.analysis_apiclient.CommunicationError` (*msg=None*, *error=None*)
Contacting Malscape failed.

class `llapi_client.analysis_apiclient.WaitResultTimeout` (*msg='Waiting for results timed out'*)
Waiting for results timed out.

5.1.6 Specifying custom command line arguments

This part of the documentation has been moved to a dedicated section, see *Application Bundle Module*.

5.1.7 Replaying traffic of pcaps for web analyses

This part of the documentation has been moved to a dedicated section, see *Application Bundle Module*.

5.2 Analysis Client Shell

The Analysis Client Shell allows running the client from the command line. This provides an interactive shell for manually sending requests to the Lastline Analyst API, and it can be used to experiment with the API for analyzing files or URLs.

This client shell is available for download at `analysis_apiclient_shell.py`.

To start the shell, place the file in the same directory as the *Analyst API client* and invoke:

```
python analysis_apiclient_shell.py <API_KEY> <API_TOKEN>
```

replacing <API_KEY> and <API_TOKEN> with your API credentials. Once the shell is started, the current context contains an analysis object. This is an instance of `llapi_client.analysis_apiclient.AnalysisClient`, which can be used to access the functionality of the Lastline Analyst API.

By default, the client connects to an API instance running in the hosted Lastline datacenters at <https://analysis.lastline.com>. To connect to a different instance, for example when using a Lastline On-Premises installation, please use the `--api-url` parameter to point to the *URL of the On-Premises API*. For example, to connect to a Lastline Analyst On-Premises running at `analyst.lastline.local`, use:

```
python analysis_apiclient_shell.py --api-url https://analyst.lastline.local/ <API_KEY>
↪ <API_TOKEN>
```

5.3 Analyst API Shell Example

This is an example of how the Analyst API shell can be used to analyze resources with the Lastline Analyst API:

```
$ python analysis_apiclient_shell.py XXXXXXXXXXXXXXXXXXXX yyyyyyyyyyyyyyyyyyyyyyy
-----
Lastline Analyst API shell
-----

[...]

In [1]:
```

Here, XXX and yyy need to be replaced with your API key and token respectively.

IPython features make the API shell easy to discover. For instance, you can use tab-autocompletion to list the methods of an object:

```
In [1]: analysis.

analysis.DATETIME_FMT          analysis.get_completed          analysis.
↪submit_exe_file
analysis.DATE_FMT              analysis.get_completed_with_metadata analysis.
↪submit_exe_hash
analysis.ERRORS                analysis.get_progress           analysis.
↪submit_file
analysis.FORMATS               analysis.get_result             analysis.
↪submit_file_hash
analysis.SUB_APIS              analysis.get_result_artifact    analysis.
↪submit_file_metadata
analysis.analyze_sandbox_result analysis.get_result_summary      analysis.
↪submit_url
analysis.completed             analysis.rescore_task           analysis.set_
↪key
```

And you can use the question-mark character after an object or method to get documentation for it:

```
In [2]: analysis.submit_file?

...
```

```
File:      analysis_apiclient.py
Definition: analysis.submit_file(self, file_stream, download_ip=None, download_
↳port=None, download_url=None, download_host=None, download_path=None, download_
↳agent=None, download_referer=None, download_request=None, full_report_score=None,
↳bypass_cache=False, delete_after_analysis=False, backend=None, raw=False,
↳verify=True)
```

Docstring:

Submit a file by uploading it.

For return values and error codes please

see `:py:meth:`malscape.api.views.analysis.submit_file``.

If there is an error and ``raw`` is not set,

a `:py:class:`AnalysisAPIError`` exception will be raised.

```
:param file_stream: file-like object containing
    the file to upload.
:param download_ip: ASCII dotted-quad representation of the IP address
    from which the file has been downloaded
:param download_port: integer representation of the port number
    from which the file has been downloaded
:param download_url: DEPRECATED! replaced by the download_host
    and download_path parameters
:param download_host: host from which the submitted file
    was originally downloaded, as a string of bytes (not unicode)
:param download_path: host path from which the submitted file
    was originally downloaded, as a string of bytes (not unicode)
:param download_agent: HTTP user-agent header that was used
    when the submitted file was originally downloaded,
    as a string of bytes (not unicode)
:param download_referer: HTTP referer header that was used
    when the submitted file was originally downloaded,
    as a string of bytes (not unicode)
:param download_request: full HTTP request with
    which the submitted file was originally downloaded,
    as a string of bytes (not unicode)
:param full_report_score: if set, this value (between -1 and 101)
    determines starting at which scores a full report is returned.
    -1 and 101 indicate "never return full report";
    0 indicates "return full report at all times"
:param bypass_cache: if True, the API will not serve a cached
    result. NOTE: This requires special privileges. Further, this
    only bypasses the MalScape caching and might still serve a
    cached result from the analysis engine
:param delete_after_analysis: if True, the backend will delete the
    file after analysis is done (and noone previously submitted
    this file with this flag set)
:param backend: DEPRECATED! Don't use
:param verify: if False, disable SSL-certificate verification
:param raw: if True, return the raw JSON results of the API query
```

The same documentation is also available [above](#).

Let's see how we can submit a URL:

```
In [3]: r=analysis.submit_url("http://www.google.com")
```

```
In [4]: r
```



```
Out [4]: {'data': {'task_uuid': 'a553401f249f4c209541799c3ccede23'}, 'success': 1}
uuid=r["data"]["task_uuid"]
```

We now have a UUID for the submitted analysis task. We can use this to request the results:

```
In [5]: result=analysis.get_result(uuid)

In [6]: result["data"]["score"]
Out [6]: 0

In [7]: import pprint
In [8]: pprint.pprint(result["data"]["report"])
{'analysis': {'artifacts': None,
              'et_results': None,
              'exploits': None,
              'network': {'redirects': None,
                          'requests': [{'content_md5': u
↳ '6294aa8e1853ee626776e7c8c9e6ff3b',
                                       'content_sha1': None,
                                       'content_type': u'text/html',
                                       'ip': None,
                                       'parent_url': u'USER_URL',
                                       'relation_type': u'6',
                                       'status': u'302',
                                       'url': u'http://www.google.com/'},
                                       {u'content_md5': u
↳ 'd0d9aabed132ac09e529453b7e26b864',
                                       'content_sha1': u
↳ 'b680dab114a88277cd7364dbdc06bac358fa0eb8',
                                       'content_type': u'text/html',
                                       'ip': u'74.125.224.191',
                                       'parent_url': u'http://www.google.com/',
                                       'relation_type': u'4',
                                       'status': u'200',
                                       'url': u'http://www.google.co.uk/'},
                                       {u'content_md5': u
↳ 'cceaec1808de4b7c9e7a0df5c213db74',
                                       'content_sha1': u
↳ 'd5c0854c026fd1ed9b15b32bae87dc73723cbd6e',
                                       'content_type': u'text/javascript',
                                       'ip': u'74.125.224.191',
                                       'parent_url': u'http://www.google.co.uk/',
                                       'relation_type': u'1',
                                       'status': u'200',
                                       'url': u'http://www.google.co.uk/xjs/_/js/
↳ k=xjs.hp.en_US.vpRA07Px3nw.O/m=sb_he,pcc/rt=j/d=1/sv=1/
↳ rs=AItrSTM2fR5rzErXQEGclenaRaSY3BLhMw'},
                                       {u'content_md5': u
↳ 'a0af21c60b0dddc27b96d9294b7d5d8f',
                                       'content_sha1': u
↳ '88ae1a2683797b31dc07d7128d7a24a628bdf52e',
                                       'content_type': u'text/javascript',
                                       'ip': u'74.125.239.15',
                                       'parent_url': u'http://www.google.co.uk/',
                                       'relation_type': u'1',
                                       'status': u'200',
                                       'url': u'http://ssl.gstatic.com/gb/js/sem_
↳ a0af21c60b0dddc27b96d9294b7d5d8f.js'}}}]},
```

```

    u'plugins': None,
    u'result': {u'analysis_ended': u'2013-07-09 04:44:51+0000',
               u'classification': u'benign',
               u'detector': u'2.6',
               u'explanation': u'models:0.84:0.00:0.84:0.00:0.00'},
    u'shellcodes': None,
    u'signatures': None,
    u'subject': {u'type': u'url',
                 u'url': u'http://www.google.com/'},
    u'threats': None},
...

```

The score is 0 since this is a benign site. The report contains more information.

Submitting an executable for analysis is similar, except that we can first check to see if the executable is already known by submitting a hash:

```

In [21]: f=open('example.file')

In [22]: import hashlib

In [23]: md5=hashlib.md5(f.read()).hexdigest()

In [24]: md5
Out[24]: 'f19e59513a23e676495fa72bd97995f2'

In [25]: analysis.submit_file_hash(md5=md5)

-----
AnalysisAPIError                                Traceback (most recent call last)

<ipython console> in <module>()

analysis_apiclient.py in submit_file_hash(self, md5, sha1, download_url, download_
->request, download_referer, full_report_score, bypass_cache, raw)
    215     purge_none(files)
    216     purge_none(params)
--> 217     return self._api_request(url, params, files=files, post=True, raw=raw)
    218
    219     def submit_file(self,

analysis_apiclient.py in _api_request(self, url, params, files, timeout, post, raw, _
->requested_format)
    423     response.raise_for_status()
    424     page = response.text
--> 425     return self._process_response_page(page, raw, requested_format)
    426
    427     def init_shell(banner):

analysis_apiclient.py in _process_response_page(self, page, raw, requested_format)
    362     else:
    363         error_code = result.get('error_code', None)
--> 364         raise AnalysisAPIError(result['error'], error_code)
    365
    366     class AnalysisClient(AnalysisClientBase):

AnalysisAPIError: Analysis API error (101): No file found matching requested hash.

```

In this case, the hash was not previously known, so an exception is raised. We should instead submit the file:

```
In [26]: f.seek(0)

In [27]: analysis.submit_file(f)
Out[27]: {'data': {'task_uuid': u'1483924fa75c440ab5493b1557ab57c5'}, u'success': 1}
```

5.4 Analyst API Shell Helpers

The API client module provides helper classes for submitting artifacts and waiting for analysis results:

```
In [28]: import analysis_apiclient
In [29]: helper = analysis_apiclient.SubmissionHelper(analysis)

In [30]: ts = helper.get_api_utc_timestamp()

In [31]: result = helper.submit_url('https://www.lastline.com/')
Submitting URL https://www.lastline.com/

In [32]: result.is_complete()
Out[32]: False

In [33]: helper.wait_for_completion_of_submission(result, ts)
Waiting for completion of 1/1 submissions
Waiting for completion of 1 submissions
Got result for task 86a5fe607aaa4181867ed9a71bcab664
Got result for task 86a5fe607aaa4181867ed9a71bcab664: AnalysisTask_
↳86a5fe607aaa4181867ed9a71bcab664(score: 1): URL=https://www.lastline.com/
Done waiting for completion of 1 submissions

In [34]: result.is_complete()
Out[34]: True

In [35]: result.task_uuid
Out[35]: u'86a5fe607aaa4181867ed9a71bcab664'

In [36]: result.score
Out[36]: 1

In [37]: result = helper.submit_filename('./myfile.exe')

In [38]: result.is_complete()
Out[38]: True

In [39]: result.score
Out[39]: 95

In [40]: helper.wait_for_completion_of_submission(result, ts)
No need to wait for completion for any of 1 submissions
```

These helper classes also support submitting multiple artifacts at once:

```
In [41]: results = helper.submit_urls_and_wait_for_completion(['http://www.google.com
↳', 'https://www.lastline.com'])
Submitting 2 URLs
Submitting URL http://www.google.com
```

```

Submitting URL https://www.lastline.com
Waiting for completion of 0/2 submissions
Done waiting for completion of 2 submissions

In [42]: results['https://www.lastline.com'].task_uuid
Out[42]: u'0ea874bab6ac4c58b374087cc0047cfb'

In [43]: results['https://www.lastline.com'].score
Out[43]: 1

In [44]: results = helper.submit_filenames_and_wait_for_completion([ './sample.exe',
↳ './hello.exe' ], bypass_cache=True)
Submitting 2 files
Submitting file hello.exe (md5=ac8545103e85219d7ff98bcd5f5a12ff,
↳ sha1=969f7183dc81eb7e95e47f06d640623be4f5ed9f)
Submitting file by hash failed: Analysis API error (101): No file found matching
↳ requested hash.
Submitting file sample.exe (md5=e7742a9e48739ce3abb686c2c1299690,
↳ sha1=b58cd7d734f9b8deb93c0f52bec8437c3c1b99ca)
Submitting file by hash failed: Analysis API error (101): No file found matching
↳ requested hash.
Waiting for completion of 2/2 submissions
[...]
Got result for task dfde1ddce80b48488034bcaaf29bc6ab
Got result for task dfde1ddce80b48488034bcaaf29bc6ab: AnalysisTask
↳ dfde1ddce80b48488034bcaaf29bc6ab: MD5=ac8545103e85219d7ff98bcd5f5a12ff,
↳ SHA1=969f7183dc81eb7e95e47f06d640623be4f5ed9f, name=hello.exe
Waiting for completion of 1 submissions
Got result for task b735251ecd44475cad34fb6a20a1d4cf
Got result for task b735251ecd44475cad34fb6a20a1d4cf: AnalysisTask
↳ b735251ecd44475cad34fb6a20a1d4cf(score: 19): MD5=e7742a9e48739ce3abb686c2c1299690,
↳ SHA1=b58cd7d734f9b8deb93c0f52bec8437c3c1b99ca, name=sample.exe
Done waiting for completion of 2 submissions

In [45]: results['./sample.exe'].task_uuid
Out[45]: u'b735251ecd44475cad34fb6a20a1d4cf'

In [46]: results['./sample.exe'].score
Out[46]: 19

```

Furthermore, the API shell also provides helper classes for accessing analysis data, such as the primary analysis subject. To download the file using the task UUID (assuming it is available in the system), use:

```

In [1]: import analysis_apiclient
In [2]: helper = analysis_apiclient.QueryHelper(analysis)

In [3]: file_stream = helper.download_analysis_subject_file(task_uuid=<task-UUID>)
In [4]: if file_stream:
...:     print 'Download successful, downloaded file has SHA1',
...:     print analysis_apiclient.hash_stream(file_stream, 'sha1')
Download successful, downloaded file has SHA1 <analysis-subject-SHA1>

```

Similarly, if the file hash of the analysis subject is known, the file can be downloaded using:

```

In [1]: import analysis_apiclient
In [2]: helper = analysis_apiclient.QueryHelper(analysis)

```

```
In [3]: file_stream = helper.download_analysis_subject_by_file_hash(sha1=<sha1>)
In [4]: with open(<output-filename>, 'wb') as outf:
...:     outf.write(file_stream.read())
```

Note that these helper classes are automatically instantiated and provided to the user when using the API client shell `analysis_apiclient_shell.py`:

```
$ python analysis_apiclient_shell.py <key> <token>

-----
Lastline Analyst API shell
-----

[...]

In [1]: result = submission_helper.submit_url('https://www.lastline.com/')
In [2]: file_stream = query_helper.download_analysis_subject_by_file_hash(md5=<md5>)
```

5.5 Application Bundle Module

The Lastline Analyst API allows detailed customization of the analysis environment via so-called *application bundles*. These bundles consist of one or more artifacts plus analysis metadata that describes the analysis environment and how to invoke the analysis subject.

The analysis system currently supports three types of bundles:

- *Program bundles* allow specifying the exact command line, with which the analysis is invoked.
- *Web replay bundles* allow replaying captured network traffic during the analysis of a web subject.
- *Document bundles* allow customizing the way Microsoft Office opens a document and allows providing a set of password candidates. While the module allows creating this type of bundles manually, it is more convenient to set the *password_candidates* parameter as part of the file submission in a call to `submit_file()` to achieve the same result.

A Python module for creating bundles is [available for download as ZIP archive](#) and contains files

- `__init__.py`
- `bundle.py`
- `helper.py`

which can be used via the command-line tool `lastline_appbundle_create.py`.

5.5.1 Specifying custom command line arguments

The API allows a user to specify command line arguments or to load/invoke an analysis subject through another application. This can be achieved using *application bundles*.

To create a simple application bundle, use the command-line tool:

```
$ python lastline_appbundle_create.py \
  -a /path/to/subject 'C:\virtual\path\to\subject.exe' \
  -- 'C:\virtual\path\to\subject.exe' arg1 arg2
```

The command-line tool provides the same options as the Python helper module, but the latter can be used directly to automate application bundle creation when using the Python API client:

```
import logging
import llappbundle.helper

# Create a logger object to be used by the appbundle creator
log = logging.getLogger()

# In all examples, the output of the create_appbundle function is a
# binary stream like object, similar to what you get when using the 'open' function
# filenames must be utf-8 encodable str type.

# Example 1: Launch an application in a specific folder
#
# To launch a program with a specific working directory, provide the
# * "files" parameter (providing a dictionary), mapping virtual filenames (filenames
# inside the analysis system) to the file-content (a local file-stream),
# * "main_subject" parameter to specify the virtual filename of the main analysis
# subject (one of the elements in the "files" dictionary), and the
# * "run_directory" parameter (optional) to specify the working directory in which
# to launch the analysis.
example1 = llappbundle.helper.create_appbundle(
    files={r"C:\virtual\path\to\subject.exe": open("/path/to/subject", 'rb')},
    main_subject=r"C:\virtual\path\to\subject.exe",
    run_directory=r"C:\path\to\run\in",
    logger=log)

# Example 2: Launch an application with specific command-line arguments.
#
# To launch a program with specific arguments, provide the
# * "files" parameter (providing a dictionary), mapping virtual filenames (filenames
# inside the analysis system) to the file-content (a local file-stream),
# * "main_subject" parameter to specify the virtual filename of the main analysis
# subject (one of the elements in the "files" dictionary),
# * "run_directory" parameter (optional) to specify the working directory in which
# to launch the analysis,
# * "executable" parameter, pointing to the main_subject, and the
# * "arguments" parameter specifying the arguments to pass to the analysis subject.
example2 = llappbundle.helper.create_appbundle(
    files={r"C:\virtual\path\to\subject.exe": open("/path/to/subject", 'rb')},
    main_subject=r"C:\virtual\path\to\subject.exe",
    executable=r"C:\virtual\path\to\subject.exe",
    arguments=["--arg1", "--arg2"],
    logger=log)

# Example 3: Start the analysis subject via the command shell (cmd.exe).
#
# To launch a program via cmd.exe, provide the
# * "files" parameter (providing a dictionary), mapping virtual filenames (filenames
# inside the analysis system) to the file-content (a local file-stream),
# * "main_subject" parameter to specify the virtual filename of the main analysis
# subject (one of the elements in the "files" dictionary),
# * "run_directory" parameter (optional) to specify the working directory in which
# to launch the analysis,
# * "executable" parameter pointing to cmd.exe, and the
# * "arguments" parameter specifying the arguments to pass to executable ("cmd.exe").
example3 = llappbundle.helper.create_appbundle(
    files={r"C:\virtual\path\to\subject.exe": open("/path/to/subject", 'rb')},
```

```

main_subject=r"C:\virtual\path\to\subject.exe",
executable=r"C:\Windows\system32\cmd.exe",
arguments=["/c", "C:\\virtual\\path\\to\\subject.exe", "arg"],
logger=log)

# Example 4: Invoke a specific function inside a DLL (via rundll32.exe).
#
# To launch a program via rundll32.exe, provide the
# * "files" parameter (providing a dictionary), mapping virtual filenames (filenames
# inside the analysis system) to the file-content (a local file-stream),
# * "main_subject" parameter to specify the virtual filename of the main analysis
# subject (one of the elements in the "files" dictionary),
# * "executable" parameter, and use "rundll32.exe" to indicate that this is the
# program to launch, and the
# * "arguments" parameter specifying the command-line for rundll32.exe, providing
# the function to invoke/use as entrypoint ("function1") as well as the arguments
# to pass to the function ("arg1" and "arg2").
example4 = llappbundle.helper.create_appbundle(
    files={r"C:\virtual\path\subject.dll": open("/path/to/subject", 'rb')},
    main_subject=r"C:\virtual\path\subject.dll",
    executable=r"C:\Windows\system32\rundll32.exe",
    arguments=[r"C:\virtual\path\subject.dll,function1", "arg1", "arg2"],
    logger=log)

# Example 5: Provide additional files
#
# To launch a program and provide an additional file as parameter, provide the
# * "files" parameter (providing a dictionary), mapping virtual filenames (filenames
# inside the analysis system) to the file-content (a local file-stream),
# * "main_subject" parameter to specify the virtual filename of the main analysis
# subject (one of the elements in the "files" dictionary),
# * "executable" parameter. As above, the template could be used but it can also be
# defined as full path since we specified it with *filename*
# * "arguments" parameter specifying the arguments to pass to the analysis subject,
# and the
# * "additional_files" parameter specifying additional files that should be copied
# to the analysis environment
example5 = llappbundle.helper.create_appbundle(
    files={
        r"C:\virtual\path\to\subject.exe": open("/path/to/subject", 'rb'),
        r"C:\somewhere\else.ini": open("/path/to/config", 'rb')
    },
    executable=r"C:\virtual\path\to\subject.exe",
    main_subject=r"C:\virtual\path\to\subject.exe",
    arguments=["--read", "C:\\somewhere\\else.ini"],
    logger=log)

# Example 6: A WScript file set as main subject launching it with wscript.exe
#
# In this example we have the following set up:
# * "files" parameter (providing a dictionary), mapping virtual filenames (filenames
# inside the analysis system) to the file-content (a local file-stream),
# * "test.vbs" is the VBS file we want to analyze
# * "executable" is the name of the executable that will launch the main subject
# * "main_subject" parameter to specify the virtual filename of the main analysis
# subject (one of the elements in the "files" dictionary),
# * "arguments" is the arguments that will be used when launching the executable
# in the system

```

```

example6 = llappbundle.helper.create_appbundle(
    files={r'test.vbs': open('test.vbs', 'r')},
    main_subject=r'test.vbs',
    executable=r'C:\Windows\system32\wscript.exe',
    arguments=[r'test.vbs'],
)

# Example 7: Setting environment variables in the system that will run the analysis
# of the bundle
#
# In this example we have the following set up:
# * "files" parameter (providing a dictionary), mapping virtual filenames (filenames
# inside the analysis system) to the file-content (a local file-stream),
# * "test.vbs" is the VBS file we want to analyze
# * "executable" is the name of the executable that will launch the main subject
# * "main_subject" parameter to specify the virtual filename of the main analysis
# subject (one of the elements in the "files" dictionary),
# * "arguments" is the arguments that will be used when launching the executable
# in the system
# * "registry_values" is a list with registry values to set in the environment
# before running the analysis.
reg_to_set = llappbundle.RegistryValue(
    path=r'HKEY_CURRENT_USER\Environment',
    name='Lastline',
    type='REG_SZ',
    value='test'
)

example7 = llappbundle.helper.create_appbundle(
    files={
        r"test.vbs": open('test.vbs', 'r')
    },
    main_subject=r'test.vbs',
    executable=r'C:\Windows\system32\wscript.exe',
    arguments=[r'test.vbs'],
    registry_values=[reg_to_set],
)

# Example 8: Executing command in the system that will run the analysis of the
# bundle after some additional commands
#
# In this example we have the following set up:
# * "files" parameter (providing a dictionary), mapping virtual filenames (filenames
# inside the analysis system) to the file-content (a local file-stream),
# * "test.vbs" is the VBS file we want to analyze
# * "executable" is the name of the executable that will launch the main subject
# * "main_subject" parameter to specify the virtual filename of the main analysis
# subject (one of the elements in the "files" dictionary),
# * "arguments" is the arguments that will be used when launching the executable
# in the system
# * "preparation_commands" is a list with commands to run in the environment
# before running the analysis.
set_date = llappbundle.Command(
    filename=None,
    executable='C:\windows\system32\cmd.exe',
    parameters=['/c', 'date', '11-19-2019'],
    run_directory=r'C:\Windows'
)

example8 = llappbundle.helper.create_appbundle(

```



```

files={
    r"test.vbs": open('test.vbs', 'r')
},
main_subject=r'test.vbs',
executable=r'C:\Windows\system32\wscript.exe',
arguments=[r'test.vbs'],
preparation_commands=[set_date],
)

```

5.5.2 Replaying traffic of pcaps for web analysis runs

Web replay bundles combine a traffic capture file (pcap) and a web subject, i.e. an URL or a HTML/JavaScript file. When a web replay bundle is submitted, the analysis engine requests the analysis subject specified in the bundle; if the subject references any external resource (e.g., scripts, stylesheets, or images), they are extracted from the capture file (instead of fetching them from their original location). This allows one to “replay” a web session that was recorded in a pcap file. Notice that web replay bundles are handled only by the instrumented browser engine.

To create a web replay bundle, use the python helper module:

```

import logging
import llappbundle.helper

# Create a logger object to be used by the appbundle creator
log = logging.getLogger()

# In all examples, the output of the create_web_replay* functions is a
# binary stream like object, similar to what you get when using the 'open' function

# Example 1: Create an appbundle with URL subject
#
# To create an web replay appbundle, provide the
# * "files" parameter (providing a dictionary), mapping virtual filenames (filenames_
↳inside the
#   analysis system) to the file-content (a local file-stream),
# * "pcap" parameter to specify the virtual filename of the pcap (one of the elements_
↳in the
#   "files" dictionary)
# * "main_subject" parameter to specify the subject URL as a string
example1 = llappbundle.helper.create_web_replay_bundle_with_url(
    files={r"traffic.pcap": open("/path/to/pcap", "rb")},
    pcap=r"traffic.pcap",
    main_subject=r"http://example.com",
    logger=log)

# Example 2: Create an appbundle with a file subject
#
# To create an web replay appbundle, provide the
# * "files" parameter (providing a dictionary), mapping virtual filenames (filenames_
↳inside the
#   analysis system) to the file-content (a local file-stream),
# * "pcap" parameter to specify the virtual filename of the pcap (one of the elements_
↳in the
#   "files" dictionary)
# * "main_subject" parameter to specify the virtual filename of the main analysis_
↳subject (one
#   of the elements in the "files" dictionary),

```

```
example2 = llappbundle.helper.create_web_replay_bundle_with_file(  
    files={  
        r"traffic.pcap": open("/path/to/pcap", "rb"),  
        r"subject.html": open(r"/path/to/subject", "rb")},  
    pcap="traffic.pcap",  
    main_subject="subject.html",  
    logger=log)
```

PYTHON MODULE INDEX

I

`llapi_client.analysis_apiclient`, 131

m

`malscape_service.api.views.analysis`, 1

`malscape_service.api.views.authentication`,
16

A

ANALYSIS_API_AUTHENTICATION_REQUIRED (in module `malscape_service.api.views.analysis`), 46
 ANALYSIS_API_CHILD_TASK_CHAIN_TOO_DEEP (in module `malscape_service.api.views.analysis`), 46
 ANALYSIS_API_DATA_NO_LONGER_AVAILABLE (in module `malscape_service.api.views.analysis`), 46
 ANALYSIS_API_FILE_EXTRACTION_FAILED (in module `malscape_service.api.views.analysis`), 46
 ANALYSIS_API_FILE_NOT_AVAILABLE (in module `malscape_service.api.views.analysis`), 45
 ANALYSIS_API_FILE_TOO_LARGE (in module `malscape_service.api.views.analysis`), 46
 ANALYSIS_API_FILE_UPLOAD_REQUIRED (in module `malscape_service.api.views.analysis`), 45
 ANALYSIS_API_INVALID_ARTIFACT_UUID (in module `malscape_service.api.views.analysis`), 46
 ANALYSIS_API_INVALID_CREDENTIALS (in module `malscape_service.api.views.analysis`), 45
 ANALYSIS_API_INVALID_D_METADATA (in module `malscape_service.api.views.analysis`), 46
 ANALYSIS_API_INVALID_FILE_TYPE (in module `malscape_service.api.views.analysis`), 46
 ANALYSIS_API_INVALID_HASH_ALGORITHM (in module `malscape_service.api.views.analysis`), 46
 ANALYSIS_API_INVALID_REPORT_VERSION (in module `malscape_service.api.views.analysis`), 46
 ANALYSIS_API_INVALID_URL (in module `malscape_service.api.views.analysis`), 46
 ANALYSIS_API_INVALID_UUID (in module `malscape_service.api.views.analysis`), 45
 ANALYSIS_API_NO_IOC_EXTRACTABLE (in module `malscape_service.api.views.analysis`), 46
 ANALYSIS_API_NO_RESULT_FOUND (in module

`malscape_service.api.views.analysis`), 46
 ANALYSIS_API_PERMISSION_DENIED (in module `malscape_service.api.views.analysis`), 46
 ANALYSIS_API_SUBMISSION_LIMIT_EXCEEDED (in module `malscape_service.api.views.analysis`), 46
 ANALYSIS_API_TEMPORARILY_UNAVAILABLE (in module `malscape_service.api.views.analysis`), 46
 AnalysisAPIError (class in `llapi_client.analysis_apiclient`), 146
 AnalysisClient (class in `llapi_client.analysis_apiclient`), 142
 AnalysisClientBase (class in `llapi_client.analysis_apiclient`), 132

C

CommunicationError (class in `llapi_client.analysis_apiclient`), 146
 create_ioc_from_result() (in module `malscape_service.api.views.analysis`), 39

E

export_report() (in module `malscape_service.api.views.analysis`), 41

G

get_analysis_tags() (in module `malscape_service.api.views.analysis`), 40
 get_api_utc_timestamp() (in module `malscape_service.api.views.analysis`), 39
 get_child_tasks_recursively() (in module `malscape_service.api.views.analysis`), 41
 get_completed() (in module `malscape_service.api.views.analysis`), 28
 get_completed() (`llapi_client.analysis_apiclient.AnalysisClientBase` method), 140
 get_completed_exported_reports() (in module `malscape_service.api.views.analysis`), 42
 get_completed_with_metadata() (in module `malscape_service.api.views.analysis`), 29

get_completed_with_metadata()
 (llapi_client.analysis_apiclient.AnalysisClientBase
 method), 141

get_exported_report() (in module
 malscape_service.api.views.analysis), 43

get_ioc_metadata() (in module
 malscape_service.api.views.analysis), 37

get_ioc_report() (in module
 malscape_service.api.views.analysis), 38

get_network_iocs() (in module
 malscape_service.api.views.analysis), 36

get_pending() (in module
 malscape_service.api.views.analysis), 31

get_progress() (in module
 malscape_service.api.views.analysis), 32

get_progress() (llapi_client.analysis_apiclient.AnalysisClientBase
 method), 141

get_report_activities() (in module
 malscape_service.api.views.analysis), 25

get_report_artifact() (in module
 malscape_service.api.views.analysis), 27

get_report_artifact() (llapi_client.analysis_apiclient.AnalysisClientBase
 method), 140

get_result() (in module
 malscape_service.api.views.analysis), 23

get_result() (llapi_client.analysis_apiclient.AnalysisClientBase
 method), 138

get_result_activities() (in module
 malscape_service.api.views.analysis), 24

get_result_artifact() (in module
 malscape_service.api.views.analysis), 26

get_result_artifact() (llapi_client.analysis_apiclient.AnalysisClientBase
 method), 139

get_result_summary() (llapi_client.analysis_apiclient.AnalysisClientBase
 method), 139

get_results() (in module
 malscape_service.api.views.analysis), 22

get_task_metadata() (in module
 malscape_service.api.views.analysis), 33

I

is_blocked_file_hash() (in module
 malscape_service.api.views.analysis), 35

is_risky_analysis_artifact() (in module
 malscape_service.api.views.analysis), 44

is_risky_analysis_artifact()
 (llapi_client.analysis_apiclient.AnalysisClientBase
 method), 141

L

llapi_client.analysis_apiclient (module), 131

login() (in module malscape_service.api.views.authentication),
 44

M

malscape_service.api.views.analysis (module), 1, 16, 153

malscape_service.api.views.authentication (module), 15,
 16

P

ping() (in module malscape_service.api.views.authentication),
 45

Q

query_file_hash() (in module
 malscape_service.api.views.analysis), 34

query_task_artifact() (in module
 malscape_service.api.views.analysis), 36

S

SubmissionHelper (class in
 llapi_client.analysis_apiclient), 142

submit_file() (in module
 malscape_service.api.views.analysis), 17

submit_file() (llapi_client.analysis_apiclient.AnalysisClientBase
 method), 132

submit_file_hash() (llapi_client.analysis_apiclient.AnalysisClientBase
 method), 134

submit_file_stream() (llapi_client.analysis_apiclient.SubmissionHelper
 method), 142

submit_file_streams_and_wait_for_completion()
 (llapi_client.analysis_apiclient.SubmissionHelper
 method), 144

submit_filename() (llapi_client.analysis_apiclient.SubmissionHelper
 method), 143

submit_filenames_and_wait_for_completion()
 (llapi_client.analysis_apiclient.SubmissionHelper
 method), 144

submit_url() (in module
 malscape_service.api.views.analysis), 20

submit_url() (llapi_client.analysis_apiclient.AnalysisClientBase
 method), 137

submit_url() (llapi_client.analysis_apiclient.SubmissionHelper
 method), 143

submit_urls_and_wait_for_completion()
 (llapi_client.analysis_apiclient.SubmissionHelper
 method), 145

W

wait_for_completion() (llapi_client.analysis_apiclient.SubmissionHelper
 method), 145

wait_for_completion_of_submission()
 (llapi_client.analysis_apiclient.SubmissionHelper
 method), 144

WaitResultTimeout (class in
 llapi_client.analysis_apiclient), 146