(S//OC/NF) Network Operations Division CNE Operational Data Exchange Format (Codex) Specification

(S//NF) NOD Specification 005: version 1

Classified By: 2341004
Derived From: CIA NSCG MET S-06
Declassify On: 25X1, 20651231

Contents

Tables

Examples

1. (U) Overview

(S//NF) The NOD CNE Operational Data Exchange (Codex) format is a series of consistent directory structures and metadata documents which reduces the burden of parsing and tracking of collected data for data recipients. This format is an "opt-in" format in the sense that it does not describe all possible data formats or metadata records that NOD or downstream analysts are interested in; instead Codex defines a number of data types and provides a preferred format for these data types. Implementers who generate data not covered by this specification may decide on the format used for that data. Data not covered by this specification OUGHT TO reside in a custom subfolder within the output folder, the folder name can be any name not already defined in this specification.

(C//NF) An important aspect of Codex is the concept of a system "fingerprint", a concise value which allows a human to quickly determine if two sets of data were collected from the same computer system. Because it can be alerting for narrowly focused tools to collect some of the information called for in a fingerprint, Codex does not require every tool to generate a fingerprint. Codex does require, however, that tools record opportunistically the portions of a fingerprint that they collect naturally to aid correlation. Fingerprints are further discussed in section 4. Codex Fingerprint XML Format.

(S//OC/NF) This specification is classified SECRET//ORCON/NOFORN to avoid hostile Foreign Intelligence Operations, Law Enforcement, Incident Response, Reverse Engineering, or any other investigation of captured tools or techniques resulting in attribution to the United States Government or the Central Intelligence Agency. Separate from that attribution the techniques discussed here are CONFIDENTIAL//NOFORN if they can be associated with Computer Network Exploitation in general and UNCLASSIFIED//FOR OFFICIAL USE ONLY otherwise.

1.1 (U) Terminology

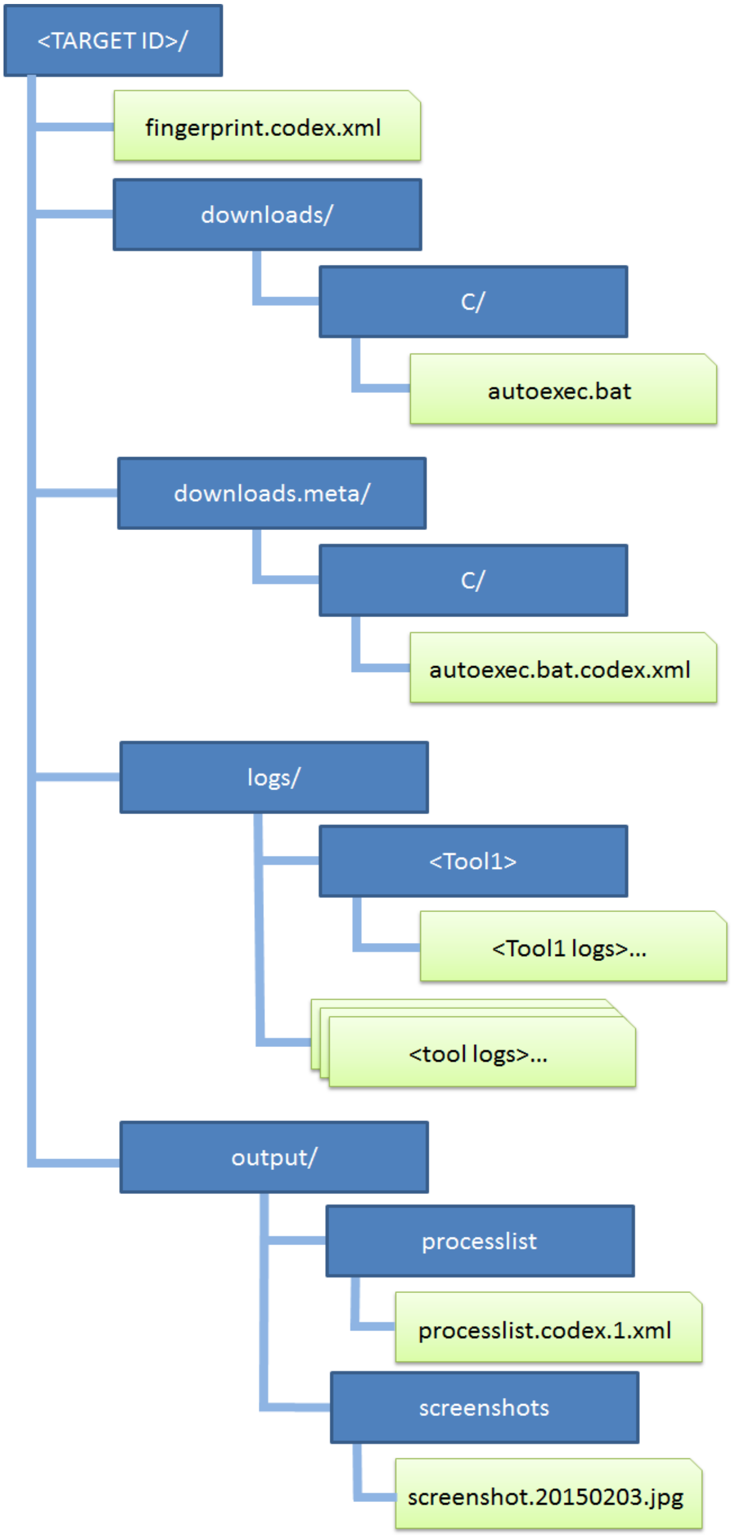
(U) The key words: MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in RFC 2119. In addition, the key words: SHOULD CONSIDER, REALLY SHOULD NOT, OUGHT TO, WOULD PROBABLY, MAY WISH TO, COULD, POSSIBLE, and MIGHT in this document are to be interpreted as described in RFC 6919.

2. (S//NF) Tradecraft Considerations

(C//NF) Diversity and lack of cross-cutting signaturability are first-order requirements for all of the sponsor's development efforts. As a result this specification is designed to be opt-in and opportunistic. This extends to the generation of the metadata specified herein. Generation of these metadata files and formats MUST NOT take place on the target. Instead, tools MUST ship back to the user the necessary details such that a Codex file or structure can be produced from the tool-specific custom format on user-controlled, potentially not internet connected, hardware. To repeat, tools MUST not store data on target in Codex format or generate any custom Codex format on target.

3. (U) Directory Structure

(U//FOUO) All Codex data will be packaged together per target entity using the following example folder structure.



(S//NF) The root folder of this structure will be named for the target. Whenever possible this folder name SHOULD reflect the target's hostname. If a tool is unable to identify the target's hostname than the root folder SHOULD be named with a tool's instance id, when a user friendly name has been defined the tool SHOULD use it rather than an opaque id. Beneath this root are several subfolders which are created as-needed: `downloads`, `downloads.meta`, `logs`, and `output`.

(U//FOUO) The `downloads` folder contains files downloaded directly from a computer in the form they appear on that computer. The `downloads.meta` folder contains metadata about such files. These are both discussed in more detail in section 5. (C//NF) Codex File Collection & Metadata.

(S//NF) The `logs` directory within the base directory will contain all tool specific logs related to operator interaction that refer to this specific system. The directory structure and format beneath this directory is not specified but SHOULD include a reference to the tool which created it. For example an interactive shell log from a computer named USER-PC this system MAY be stored as `USER-PC/logs/exampletool-examplelog.txt` or MAY be stored as `USER-PC /logs/exampletool/examplelog.txt` or MAY be stored as `USER-PC/logs/shell-examplelog.txt.`

(S//NF) The `output` directory is used for the collection of data other than files resident on a target disk. This includes any synthesized data (e.g., surveys, screenshots) or unpacked or decrypted data not directly related to an operator's interaction with the tool (e.g., target keystroke logs). Each specific Codex data type has a defined folder name where data of that type MUST be stored as a subfolder of the output directory.

(U//FOUO) The `fingerprint.codex.xml` file is described in section 4. (U//FOUO) Codex Fingerprint XML Format.

(U//FOUO) Files already present in the Codex directory structure MUST NOT be overwritten. If a new file has the same name as an existing file a monotonic counter value will be appended to the filename just before the extension (e.g., duplicate.2.xml).

(U//FOUO) Unless otherwise specified in this document all Codex specified files are UTF-8 encoded and all timestamps SHOULD be in ISO 8601 format (`YYYY-MM-DD HH:MM:SS.zzzzzzZ`) in Coordinated Universal Time (UTC). Note that in accordance with the specification and by mutual agreement the "T" separator is replaced with a space character and the sub-second values (.zzzzzz) are optional. For example, March 14th, 2015 at 8:09:26am EST would be recorded as "2015-03-14 13:09:26Z".

(U//FOUO) To avoid multiple processes overwriting each other's changes in the case of parallel execution all files SHOULD be locked for exclusive use when open for writing. To avoid potential deadlock tools MUST lock files in lexographical order based on the filename if two or more files need to be open for writing at the same time.

3.1 (U//FOUO) Codex XML Format

(U//FOUO) The Codex specification defines a common XML format that is used for storing custom metadata about different types of retrieved data. This specification is composed of independently versioned subsections that further define the format for additional types of collection data and associated metadata.

(U//FOUO) All Codex XML files MUST start with a Codex tag with a `version` attribute, the `version` attribute MUST correspond to the version of the Codex specification being used. The Codex section MUST contain at least one data tag that corresponds to a defined Codex data type. Codex data tags MUST include a `version` attribute that refers to the version number of the data type specification. The Codex section MUST also contain a `timestamp` tag; this tag indicates when the action that generated the data was taken, not when the data was retrieved. If the data was generated by multiple actions this SHOULD be the most recent applicable timestamp. If a tool wants to record additional data in a Codex XML file the tag names MUST be prefixed with `private_<tool id>_`, where `<tool id>` is a two letter unclassified abbreviation for the tool.

```
<codex version=”1”>
    <dirwalk version=”1”>
       <!--- Data Type specific format within ---!>
       <private_cf_magic>private magic</private_cf_magic>
    </dirwalk>
    <timestamp>2015-01-06 09:21:00.963000Z</timestamp>
</codex>
```

Example 1: (U//FOUO) Codex XML Example

4. (U//FOUO) Codex Fingerprint XML Format

Version: 1.0

(C//NF) Codex utilizes a collection of distinctive values to generate a loose fingerprint to attempt to uniquely and repeatably identify a particular target. Aside from the difficulty of ensuring perfect uniqueness in the face of repeatability, perfect accuracy of a fingerprint conflicts with the sponsor's need to avoid heuristic signatures and therefore the collection of the necessary information for a fingerprint is optional.

(C//NF) The information which comprises a fingerprint was chosen to enable independent tools to generate the same fingerprint when run on the same computer, barring significant environmental changes. The sponsor believes this will enable correlation and differentiation of independent collection events from the same computer or to be alerted when significant changes do occur (e.g., an installation is imaged onto a new computer).

(U//FOUO) Tools which collect the necessary information in the course of their normal behavior SHOULD record this information to assist correlation, even if they do not collect enough information to generate the full fingerprint value. If a tool attempts to obtain some of the information necessary for a fingerprint and fails for any reason then no fingerprint SHOULD be generated and the error reported to the user.

(U//FOUO) The fingerprint file is required to exist for the Codex collection to be considered complete and ready for use by a processing system. This does not mean that every implementer is responsible for ensuring the fingerprint file is complete, instead the sponsor will ensure this requirement is met by at least one tool utilized in every collection even if this means the fingerprint file is provided by the user during a final processing step.

(U//FOUO) Types of Fingerprints

(C//NF) This specification defines two primary types of targets, `machine` and `account`, representing a computing device or an individual account with some service provider respectively. Only one of either `machine` or `account` fingerprints need be provided. Each primary target type has its own fingerprint type or types. Machine targets have two unique fingerprint types: `os` and `hardware`; `account` targets only have a `user` fingerprint. Most target machines will only be identified by either an OS or a hardware fingerprint. For finalized Codex fingerprints the sponsor will be responsible for ensuring the appropriate fingerprint type is complete, based on the access methods and capabilities.

4.1 (U//FOUO) Common Fingerprint Formatting

(U//FOUO) Many fingerprint values are the MD5 hash of the string concatenation of several values. All component strings MUST be UTF-8 encoded (with no byte order marker), have all characters converted to lowercase, and all leading and trailing whitespace removed before any computation is performed. MD5 was chosen due to its use elsewhere in this specification and because no specific resistance to intentional collision is required for this application.

4.2 (U//FOUO) Machine Fingerprints

4.2.1 (U//FOUO) Operating System Instance (OS) Fingerprint

(C//NF) An OS fingerprint should uniquely identify a specific installation of an OS. Please note that collisions can still occur, especially in corporate deployment scenarios. A collision is a clue to the user that some level of volatility might be expected from the OS and that the Hardware identifier should be considered more reliable. In this way a non-unique OS fingerprint is a feature, not a bug.

(U//FOUO) For Windows the OS Fingerprint is the MD5 hash of the following template string.

```
<OS Name>-<Install Date>-<Hostname>-<Registered Owner>
```

(U//FOUO) Valid values are similar to the output of the following command:

```
wmic.exe os get version, installdate, csname, registereduser
```

(U//FOUO) For example a Windows 7 SP1 machine named USER-PC registered to "Joe User" would have an OS fingerprint of "bc89504e2794514ba593cc2934bd6b96" which is the MD5 hash of the UTF-8 string "6.1.7601-20131112211240.000000-300-user-pc-joe user".

(U//FOUO) For Linux the OS Fingerprint will be the contents of /etc/machine-id, if this file does not exist or is empty then the contents of /var/lib/dbus/machine-id will be used. If neither of these files exists or they exist but are empty then the OS Fingerprint will be the MD5 hash of the lowercase file system UUID for the root file system (i.e. '/'). For example a Linux machine with none of the above files but with a root UUID of "27a0727c-d9cb-c412-a618-9c573f9a015f" would have an OS fingerprint of "a098ec3e9cdd7183b6db428b64bdb7e0".

(U//FOUO) For Apple OSX the OS Fingerprint will be the MD5 hash of the filesystem UUID for the root file system (i.e. '/'). For example a Mac with a root UUID of "de305d54-75b4-431b-adb2-eb6b9e546013" would have an OS fingerprint of "c001163fbbaaadabeb733e1e9ceb95e6".

(U//FOUO) If no OS fingerprint can be determined despite the tool's best effort then a fingerprint UID value SHOULD NOT be generated.

4.2.2 (U//FOUO) Hardware Fingerprint

(U//FOUO) A hardware fingerprint should uniquely identify the core hardware of a target system. The Codex Hardware fingerprint is defined as the MD5 hash of the following string.

```
<Boot Drive Hardware Serial#>-<BIOS UUID>
```

(U//FOUO) Valid values are similar to the output of the following commands:

(U//FOUO) Windows:

```
wmic.exe diskdrive get serialnumber

wmic.exe csproduct get uuid
```

(U//FOUO) Linux:

```
hdparm -I /dev/sda | grep ‘Serial Number’ | awk ‘{ print $3 }’

dmidecode -s system-uuid
```

(U//FOUO) For example a computer with boot drive serial number “W -DCW2H0C073195” and BIOS UUID “412AF010-43B2-18F2-0000-C5C239B71D30” would have a Hardware Fingerprint of “820fb61122e0c0f90c01f4ac7adc57cf”.

(U//FOUO) If no Hardware fingerprint can be determined despite the tool’s best effort then a fingerprint UID value SHOULD NOT be generated.

4.3 (U//FOUO) Account Fingerprints

4.3.1 (U//FOUO) User Fingerprint

(C//NF) A user fingerprint should uniquely identify a specific account based on the user ID of the account. This is primarily intended to apply in situations where a machine fingerprint can never be obtained or would result in an incorrect or non-intuitive understanding of the source of the data (e.g., collection from a webmail account). Identification of user-level accounts on systems where a machine fingerprint can be generated is not necessary as collection from that account should be associated with the machine fingerprint as the stronger identifier. The Codex user fingerprint is defined as the MD5 hash of the following string:

```
<Service Provider>-<Full User ID>
```

(C//NF) The inclusion of Service Provider is intended to avoid overlap when multiple services use the same email address for user identification. In the case of an email account, the Service Provider would be the email provider through which the data was collected. Collection tools and users generating Codex packages are responsible for consistent use of Service Provider within an operation.   For example the fingerprint for example@example.com being accessed via a provider identified by a user as “Jmail” would be “7b24dfac979f76de77d76308027ca0a1”, the MD5 hash of “jmail-example@example.com”

4.4 (U//FOUO) Fingerprint XML Format

(U//FOUO) The fingerprint file MUST conform to the common Codex XML format and MUST use fingerprint as the root tag for its data. In addition the fingerprint file will contain the following keys and appropriate values. Note that required fields are only required to be present when the Codex directory structure is finalized and such fields may be absent during use.

Key	Required	Comments
fingerprint	Yes	Root tag for fingerprint data including target, type, and version attributes
uid	Yes	Generated fingerprint as defined for the target and type
usertag	Yes	User defined value for separating systems that have the same fingerprint
osversion	No	OS Name used in Windows OS fingerprint
installdate	No	Install Date used in Windows OS fingerprint
owner	No	Registered Owner used in Windows OS fingerprint
hostname	No	Hostname used in OS fingerprint
machineid	No	Contents of machine-id file used in Linux OS fingerprint
rootfsid	No	Root Filesystem UUID
bootdriveserial	No	Boot Drive Serial# used in Hardware fingerprint
biosuuid	No	Bios UUID used in Hardware fingerprint
serviceprovider	No	Provider of the account that is being accessed
accountid	No	Full user id used to authenticate access to the targeted account

Table 1: (U//FOUO) Fingerprint XML Keys

```
<codex version=”1”>
    <fingerprint target=”machine” type=”os” version=”1”>
       <uid>bc89504e2794514ba593cc2934bd6b96</uid>
       <osversion>6.1.7601</osversion>
       <installdate>20131112211240.000000-300</installdate>
       <owner>Joe User</owner>
       <hostname>USER-PC</hostname>
       <usertag>NONE</usertag>
    </fingerprint>
    <fingerprint target=”machine” type=”hardware” version=”1”>
       <uid>820fb61122e0c0f90c01f4ac7adc57cf</uid>
       <bootdriveserial>W -DCW2H0C073195</bootdriveserial>
       <biosuuid>412AF010-43B2-18F2-0000-C5C239B71D30</biosuuid>
       <usertag>Laptop</usertag>
    </fingerprint>
    <timestamp>2015-01-06 09:21:00.963000Z</timestamp>
</codex>
```

Example 2: (U//FOUO) Machine Fingerprint XML File

```
<codex version=”1”>
    <fingerprint target=”account” type=”user” version=”1”>
       <uid>7b24dfac979f76de77d76308027ca0a1</uid>
       <serviceprovider>Jmail</serviceprovider>
       <accountid>example@example.com</accountid>
       <usertag>NONE</usertag>
    </fingerprint>
    <timestamp>2015-01-06 09:21:00.963000Z</timestamp>
</codex>
```

Example 3: (U//FOUO) Account Fingerprint XML File

4.5 (U//FOUO) Implementation Details

(U//FOUO) This section defines additional responsibilities for all tools implementing any part of the Codex Fingerprint specification.

4.5.1 (U//FOUO) Best Effort Fingerprint Data Collection

(U//FOUO) If a tool collects data that is a component of a defined fingerprint type the tool SHOULD record that data in the fingerprint file. This allows multiple sponsor tools to work together to build a single fingerprint.

4.5.2 (U//FOUO) Fingerprint File Collaboration

(U//FOUO) Tools implementing the Codex Fingerprint specification MUST read any existing fingerprint file and add additional data rather than overwriting an existing file. If the tool detects a discrepancy between existing data and its collected data, the tool MUST notify the user and MUST NOT modify the conflicting data in the fingerprint file. After writing additional data to a fingerprint file the tool MUST check to see if there enough data to generate the UID for that fingerprint type and MUST record the UID if possible.

5. (C//NF) Codex File Collection & Metadata

Version 1.0

(C//NF) This document defines how collected files will be stored in a Codex package as well as a file format for metadata associated with file collection. Collected files are any files retrieved directly from a target's filesystem.

5.1 (C//NF) Collected Files

(C//NF) Within the base folder collected files will be stored in the `downloads` subfolder. For every file present in the `downloads` folder there MUST also be an associated metadata file in the `downloads.meta` subfolder of the base folder. Collection metadata files MUST have the same name as the collected file with `.codex.xml` appended to the name. The subfolders of the `downloads` and `downloads.meta` folders represent the path to the file on the targeted system (e.g. `C:\WINDOWS\plAns.txt` will be stored as `<targetid>/downloads/C/WINDOWS/plAns.txt` with an associated `<targetid>/downloads.meta/C/WINDOWS/plAns.txt.codex.xml`). Reconstructing the target path in this way risks encountering the MAX_PATH limitation on (non-target) Windows-based computers processing Codex data. Tools interacting with Codex data on Windows MUST understand and avoid this limitation (e.g., by utilizing UNC paths).

(C//NF) Multiple downloads of the same filename SHALL have a monotonic integer appended to the filename (e.g., `<targetid>/downloads/C/WINDOWS/plAns.txt.1` in the above example). A subsequent download of the file "C:\WINDOWS\plAns.txt.1" from target would be disambiguated by appending an additional integer (e.g., `<targetid>/downloads/C/WINDOWS/plAns.txt.1.1`). Please note that this differs from the disambiguation counter used for all other Codex files due to the uncontrollable nature of a downloaded file's name and extension.

5.2 (C//NF) Collection Metadata

(U//FOUO) The metadata file MUST conform to the common Codex XML format and MUST use a `file` tag as the root of its data. In addition the metadata file will contain the following keys and appropriate values.

Key	Required	Comments
name	YES	Original filename
dirname	YES	Original path to file, not including name
size	YES	File size in bytes
md5	NO	MD5 Hash of file contents
hash	NO	Other hash of file contents, hash type MUST be specified by the Type attribute
modifiedtime	NO	Last Modified Timestamp
accessedtime	NO	Last Accessed Timestamp
createdtime	NO	File Created Timestamp
owner	NO	File owner
group	NO	Group associated with file (if appropriate for target operating system)

Table 2: (U//FOUO) File Collection Metadata Keys

```
<codex version=”1”>
    <file version=”1”>
        <name>evilplans.txt</name>
        <dirname>C:\Windows</dirname>
        <size>1024</size>
        <md5>2cad20c19a8eb9bb11a9f76527aec9bc</md5>
        <hash type=”sha1”>dbc6f891ed1aa830aed20ccfa923cc10ca6eb0ab</hash>
        <modifiedtime>2014-12-15 15:42:12.963Z</modifiedtime>
        <accessedtime>2014-12-15 15:42:12.963Z</accessedtime>
        <createdtime>2014-12-15 15:42:12.963Z</createdtime>
        <owner>Dr. Evil</owner>
    </file>
    <timestamp>2015-01-06 09:21:00.963000Z</timestamp>
</codex>
```

Example 4: (U//FOUO) File Collection Metadata File

6. (U//FOUO) Codex Dirwalk Format

Version: 1.0

(U//FOUO) Directory walks (dirwalks) consist of two files, a data file containing the directory walk data and a metadata file containing metadata on how and when the directory walk was executed.

6.1 (U//FOUO) Directory Walk Data File

(U//FOUO) All directory walk data files SHALL be stored in a `dirwalks` subfolder within the output folder. All Directory walk data files SHALL conform to the defined naming scheme:

```
        dirwalk.<root>.<counter>.csv
```

(U//FOUO) The `root` in the naming scheme is the directory name of the directory at the root of the directory walk. For example, if the directory walk was executed on "C:\Program Files\360safe", the path and filename of the output would be "<targetid>/output/dirwalks/`dirwalk.360safe.csv`".

(U//FOUO) The `counter` in the naming scheme is a monotonic integer counter used to ensure that existing dirwalk files are not overwritten; the counter is OPTIONAL if no other dirwalk files exist (e.g. dirwalk.C.csv, dirwalk.C.1.csv, dirwalk.C.2.csv, etc.)

(U//FOUO) The dirwalk data file format is RFC 4180 CSV encoded in UTF-8. Because many CSV libraries do not handle embedded commas and newlines correctly, implementers MUST ensure that they correctly escape such values.

(U//FOUO) Fields MUST be recorded in the same order as they are documented in Table 3. For ease of processing the first record of the data file will be a list of fields using the Tag specified in Table 3. Fields that are not captured, or for some other reason have no data MUST be indicated with a blank field (i.e. two concurrent commas.) Fields may be left out of the file only if all following fields were also not captured. For example, ADS cannot be left out of the file if Owner and Group fields are captured, however if Owner, Group and Unix Permission fields are not captured, then they can be left out of the file.

Name	Tag	Required	Comments
Path	PATH	YES	Full path and name of the file (e.g. C:\windows\plans.txt)
Type	TYPE	NO	File type Accepted values are: FILE – Regular File DIR – Directory LINK – Symbolic link JUNC – A Windows Junction point or a Unix hard link DEV – A Unix block or character special file PIPE – A Unix Named Pipe (FIFO) SOCK – A Unix Socket
Size	SIZE	YES	Size of the file in bytes (e.g. 12543242)
Short Name	SNAME	NO	Short (8.3) version of the file name without path
Modified Time	MODT	NO	Last modified time
Access Time	ACCT	NO	Last accessed time
Created Time	CRET	NO	File creation time
ADS	ADS	NO	Indicates the existence of an Alternate Data Stream (ADS). Accepted values are: True or False
Owner	OWNER	NO	File's owner
Group	GROUP	NO	File's Unix group
Unix Permissions	UPERM	NO	File's Unix permission bits as octal number

Table 3: (U//FOUO) Dirwalk Data Fields

6.2 Example Directory Walk CSV Files

(U//FOUO) Below is an example dirwalk data file that captured the path, type, size, short name, modified, accessed, created timestamp fields, and ADS stream information for two files in the `C:\Evil` directory.

```
PATH,TYPE,SIZE,SNAME,MODT,ACCT,CRET,ADS\r\n
C:\Evil,DIR,0,evil,2014-12-15 15:42:12.963,2014-12-15 15:42:12.963,2014-12-15
15:42:12.963,False\r\n
C:\Evil\plans.txt,FILE,2411,plans.txt,2014-12-15 15:42:12.963,2014-12-15
15:42:12.963,False\r\n
C:\Evil\myhenchmen.txt,FILE,2411,myhenc~1.txt,2014-12-15 15:42:12.963,2014-
12-15 15:42:12.963,False\r\n
```

Example 5: (U//FOUO) Directory Walk CSV File

(U//FOUO) Below is an example dirwalk data file that captured the path, type, size, short name, and modified, accessed, created timestamp fields, and a single example record for a file named ‘C:\evil”plans”,1.txt’.

```
PATH,TYPE,SIZE,SNAME,MODT,ACCT,CRET\r\n
“C:\evil””plans””,1.txt”,FILE,1024,evilpl~1.txt,2014-12-15 15:42:12.963,2014-
12-15 15:42:12.963,2014-12-15 15:42:12.963\r\n
```

Example 6: (U//FOUO) Directory Walk CSV File with Complex Characters

6.3 (U//FOUO) Directory Walk Metadata Format

(U//FOUO) All directory walk metadata files MUST be stored in the same location as the directory walk data file, and MUST have the same name as the directory walk data file with `.codex.xml` appended to the name, for example `dirwalk.C.1.csv.codex.xml`. The metadata file MUST contain conform to the common Codex XML format and MUST use a `dirwalk` tag as the root of its data. In addition the metadata file will contain the following keys and appropriate values:

Key	Required	Comments
isfiltered	YES	“True” if the directory list is not a full and comprehensive walk of root, otherwise “False”
root	YES	The root directory or drive of the directory walk
recursive	YES	“True” if the directory list continued into all child folders, otherwise “False”
recursivedepth	YES, if recursive is True	Levels of child folders that the directory list included, where 0 means that no child folders were listed and -1 means that every level was listed
filter	NO	Tool specific filter syntax that was applied

Table 4: (U//FOUO) Directory Walk Metadata XML Keys

```
<codex version=”1”>
    <dirwalk version=”1”>
        <isfiltered>True</isfiltered>
        <root>C:\</root>
        <recursive>True</recursive>
        <recursivedepth>5</recursivedepth>
        <filter>*.txt</filter>
    </dirwalk>
    <timestamp>2015-01-06 09:21:00.963000Z</timestamp>
</codex>
```

Example 7: (U//FOUO) Directory Walk Metadata File

7. (U//FOUO) Process List Format

Version 1.0

(U//FOUO) The Process List file SHALL be stored in a `processlist` subfolder within the output folder. All process list files SHALL conform to the defined naming scheme:

```
      processlist.codex.<counter>.xml
```

(U//FOUO) Where `counter` is a monotonic increasing integer used to prevent new process list data from overwriting an existing file. The process list file MUST conform to the common Codex XML format and MUST use a `processlist` tag as the root of its data. In addition the process list file will contain the following keys and appropriate values.

Key	Required	Comments
process	YES	Represents a single process.
pid	YES	The OS assigned Process ID
imagename	YES	The Process's image name
parentpid	NO	The process parent's Process ID
bitness	NO	"32" if the process is a 32 bit process, "64" if the process is a 64 bit process, or "NONE" if the bitness is unknown or other
session	NO	The Windows session number or "NONE" if unknown
libraries	NO	A container for loaded libraries represented as `<library>` tags
library	NO	A specific loaded library or "NONE" if unknown
commandline	NO	The command line arguments provided to the process or "NONE" if unknown
environment	NO	A container representing the process's environment or "NONE" if unknown
envvar	NO	A specific environment variable in key=value format or "NONE" if unknown
user	NO	The user the process is running as or "NONE" if unknown
memoryused	NO	The amount of resident memory in bytes or "NONE" if unknown
startuptime	NO	The timestamp of the process's start time or "NONE" if unknown
openfiles	NO	A container for files opened by the process at the time of the process listing
file	NO	The filename of a specific open file or "NONE" if unknown
isdebugged	NO	"True" if the process is currently attached to a userspace debugger, "False" if it is known to not be attached, "NONE" if unknown

Table 5: (U//FOUO) Process List XML Keys

```
<codex version=”1”>
    <processlist version=”1”>
        <process>
            <pid>1124</pid>
            <imagename>calc.exe</imagename>
            <parentpid>514</parentpid>
            <bitness>32</bitness>
            <session>1</session>
            <libraries>
                <library>ntdll.dll</library>
            </libraries>
            <commandline>c:\WINDOWS\System32\calc.exe</commandline>
            <environment>
                <envvar>windir=C:\WINDOWS</envvar>
            </environment>
            <user>USER-PC\Administrator</user>
            <memoryused>610304</memoryused>
            <startuptime>2015-01-06 01:09:12Z</startuptime>
            <openfiles>
                <file>c:\WINDOWS\System32\calc.exe</file>
            </openfiles>
            <isdebugged>False</isdebugged>
        </process>
    </processlist>
    <timestamp>2015-01-06 09:21:00.963000Z</timestamp>
</codex>
```

Example 8: (U//FOUO) Process List Collection File

8. (C//NF) Screenshot Collection Format

Version 1.0

(U//FOUO) Screenshots SHALL be stored in the `screenshots` subfolder within the output directory. All screenshots SHALL conform to the defined naming scheme:

```
    screenshot.<YYYYMMDD-HHMMSS>.<toolspecific>.<ext>
```

(U//FOUO) Where `<ext>` is the appropriate extension for the image format, `<YYYYMMDD-HHMMSS>` is the timestamp referring to the time the screenshot was taken, and `<toolspecific>` is a free-form value the particular tool may use to provide additional context or disambiguation for each particular image.

(U//FOUO) For every screenshot present there MUST also be an associated metadata file of the same name as the collection file with `.codex.xml` appended to the name. For example a screenshot named "`screenshot.20150106-092133.calc.jpg`" would have a metadata file named "`screenshot.20150106-092133.calc.jpg.codex.xml`". The metadata file MUST conform to the common Codex XML format and MUST use a `screenshot` tag as the root of its data. In addition the metadata file will contain the following keys and appropriate values.

Key	Required	Comments
originalsize	YES	The original window's size in "WxH" where W is the number of horizontal pixels and H is the number of vertical pixels
isfullscreen	YES	"True" if the screenshot was taken of the entire screen, otherwise "False"
isminimized	NO	"True" if the subject of the screenshot was minimized at the time the screenshot was taken, otherwise "False"
activewindow	NO	"True" if the subject of the screenshot was in the foreground at the time the screenshot was taken, otherwise "False"
pid	NO	The subject window's associated Process ID or "NONE" if unknown or inapplicable
imagename	NO	The subject window's associated image name or "NONE" if unknown or inapplicable
windowtitle	NO	The subject window's title or "NONE" if unknown or inapplicable
session	NO	The subject window's session or "NONE" if unknown or inapplicable
user	NO	The subject window's user or "NONE" if unknown or inapplicable
monitor	NO	The screenshot subject's monitor number

Table 6: (U//FOUO) Screenshot Collection Metadata Keys

```
<codex version=”1”>
    <screenshot version=”1”>
        <originalsize>680x320</originalsize>
        <isfullscreen>False</isfullscreen>
        <isminimized>False</isminimized>
        <activewindow>True</activewindow>
        <pid>1364</pid>
        <imagename>c:\WINDOWS\System32\calc.exe</imagename>
        <windowtitle>Calculator</windowtitle>
        <session>1</session>
        <user>USER-PC\Administrator</user>
        <monitor>1</monitor>
    </screenshot>
    <timestamp>2015-01-06 09:21:00.963000Z</timestamp>
</codex>
```

Example 9: (U//FOUO) Screenshot Metadata File

9. (C//NF) Email Collection Format

Version: 1.0

(C//NF) This document defines a file format for capturing collected email messages and associated metadata from a targeted system or targeted account. All files related to the email collection will be stored in an `email` subfolder within the output folder. Within the `email` subfolder there SHALL be a folder for each account that was collected. The account subfolder MUST have the same name as the account that was collected (e.g. `example@example.net/`). For each account that was collected there MUST be an account collection metadata file for storing information. The account collection metadata file MUST be stored in the same folder as the account folder, and MUST have the same name with `.codex.xml` appended, for example, `example@example.net.codex.xml`.

9.1. (C//NF) Account Collection Metadata

(U//FOUO) The metadata file for each collected account MUST conform to the common Codex XML format and MUST use an `email` tag as the root of its data. In addition the metadata file will contain the following keys and appropriate values:

Key	Required	Comments
account	YES	Account name being collected (e.g., example@example.net)
status	YES	Final status of the attempt. COMPLETE –finished successfully PARTIAL –terminated before being completed FAILED –failed, no email collected
requestedstart	YES	If this attempt was filtered by time window, the start of the time window (i.e., emails after this timestamp). This is the requested value, regardless of status (i.e., partial or failed should not affect this value). If no filter was applied this value SHOULD be NONE
requestedend	YES	If this attempt was filtered by time window, the end of that time window (i.e., emails before this timestamp). This is the requested value, regardless of status (i.e., partial should not affect this value). If no filter was applied this value SHOULD be NONE

Table 7: (U//FOUO) Email Metadata Keys

```
<codex version=”1”>
    <email version=”1”>
        <account>example@example.net</name>
        <status>COMPLETE</path>
        <requestedstart>2014-12-15 15:42:12.963Z</requestedstart>
        <requestedend>NONE</requestedend>
    </email>
    <timestamp>2015-01-06 09:21:00.963000Z</timestamp>
</codex>
```

Example 10: (U//FOUO) Email Account Metadata File

9.2. (C//NF) Collected Email Messages

(U//FOUO) Each individual message MUST be stored as an EML file (as commonly defined by RFC 822) conforming to the following naming scheme.

```
<Message-ID>-<FLAGS>.eml
```

(U//FOUO) Message-ID is the value of the Message-ID field as defined in RFC 822. FLAGS is a collection of characters representing any of the following possible flags related to the message which are set on the server. The recording of these flags SHOULD be done by any tool retrieving this data, but may be limited by the retrieval mechanism. Flag ordering is not significant. An uppercase flag letter code indicates that the flag has been checked and is True, a lowercase flag letter code indicates that the flag has been checked and is False, and the special letter code “N” indicates that no flag checking was performed. In all cases at least one flag MUST be provided.

Flag	Letter Code	Comments
Read	R	The message has been marked as read by the user
Answered	A	The message is marked as having a reply being sent
Draft	P	This message is marked as a draft
Deleted	D	This message has been marked as deleted, but not yet purged from the system
Header	H	This message only contains header data and no content. This flag is influenced by the retrieval mechanism rather than the server’s state.
No flags	N	Flag data was not retrieved due to tool or service limitation. This flag is influenced by the retrieval mechanism rather than the server’s state.

Table 8: (U//FOUO) Email Message Flags

(C//NF) Each EML file SHALL be stored in the account subfolder replicating the folder structure of the account that was collected. For example a read and replied-to email collected from `evil@drevil.net` with the message-id of `DEADBEEF` that was located in the `INBOX/Plans` folder would be stored as:

```
<targetid>/output/email/evil@drevil.net/INBOX/Plans/DEADBEEF-RA.eml
```

10. (U//FOUO) Netstat File Format

Version 1.0

(U//FOUO) The Network connection statistics File (netstat file) SHALL be stored in a `netstat` subfolder within the output folder. All netstat files will conform to the defined naming scheme:

```
    netstat.codex.<counter>.xml
```

(U//FOUO) Where counter is a monotonic increasing integer used to prevent new netstat data from overwriting an existing file. The netstat file MUST conform to the common Codex XML format and MUST use a `netstat tag` as the root of its data. In addition the netstat file MUST contain the following keys and appropriate values inside the data section. Local in this context refers to the machine the netstat was run on.

Key	Required	Comments
connection	YES	Represents a single connection
localip	YES	Connection's local IP address relative to the examined computer
localport	YES	Connection's local port relative to the examined computer
protocol	YES	Connection Layer 4 protocol (e.g., TCP, UDP)
state	YES	State of connection or "NONE" if unknown or stateless
remoteip	YES	Connection's remote IP relative to the examined computer. Value SHOULD be "NONE" if this is a listening Socket, or remote IP cannot be determined
remoteport	YES	Connection's remote port relative to the examined computer. Value SHOULD be "NONE" if this is a listening Socket, or remote port cannot be determined
ourtraffic	NO	True if this connection was generated by the tool retrieving this data, otherwise False
pid	NO	PID associated with this connection
imagename	NO	Image name of process associated with this connection

Table 9: (U//FOUO) Netstat Collection XML Keys

```
<codex version=”1”>
    <netstat version=”1”>
        <connection>
            <localip>10.0.0.1</localip>
            <localport>1642</localport>
            <protocol>TCP</protocol>
            <state>ESTABLISHED</state>
            <remoteip>10.0.0.2</remoteip>
            <remoteport>445</remoteport>
            <ourtraffic>True</ourtraffic>
            <pid>1340</pid>
            <imagename>NotEvil.exe</imagename>
        </connection>
        <connection>
            <localip>10.0.0.1</localip>
            <localport>1635</localport>
            <protocol>TCP</protocol>
            <state>LISTENING</state>
            <remoteip>NONE</remoteip>
            <remoteport>NONE</remoteport>
            <ourtraffic>True</ourtraffic>
            <pid>1342</pid>
            <imagename>NotEvil.exe</imagename>
        </connection>
        <connection>
            <localip>10.0.0.1</localip>
            <localport>1635</localport>
            <protocol>UDP</protocol>
            <state>NONE</state>
            <remoteip>NONE</remoteip>
            <remoteport>NONE</remoteport>
            <ourtraffic>False</ourtraffic>
            <pid>1112</pid>
            <imagename>Evil.exe</imagename>
        </connection>
    </netstat>
    <timestamp>2015-01-06 09:21:00.963000Z</timestamp>
</codex>
```

Example 11: (U//FOUO) Netstat Collection File