

APIs and CLIs for Independent Evaluations Version 1.1

June 11, 2019

What is the ActEV Evaluation CLI?

- The ActEV Evaluation CLI is a prescribed set of command line utilities that prescribe both the function of each utility and the required command line options
- It is a simplified view of installing a software system and running the system
- It is designed as an 'Abstract Command Line Interface' (see slide 3)
- It is written in Python3

What is an “Abstract Command Line Interface”?

Definition:

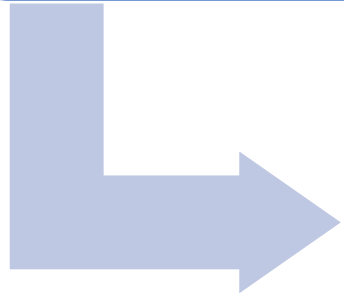
An abstract **CLI** is a generic set of command line utilities used as a basis for creating specific utilities that conform to a specific protocol, or the set of operations it supports. Abstract CLIs are not implemented directly but rather delegated to a specific implementation.

Abstract **CLIs** are useful when creating user interactions that model reality because they make it possible to specify an invariant level of functionality in the top level utilities, but leave the implementation to the developer.

* Definition adapted from a “Abstract Class” definition defined on Technopedia
<https://www.techopedia.com/definition/17408/abstract-class>

Submission Information Collection

- Specifies where to download the system submission from
- Supported tasks/activities
- System output on a validation video and activities
- Sent via the ActEV scoring server



ActEV Evaluation Command Line Interface

- Controls processing steps of an ActEV system

This slide deck's focus

ActEV Evaluation CLI Design Considerations

- Flexibly support many types of system designs:
 - Single git repo, single container, multi-container
 - Scalable for multi-node solutions in the future
- Rigorous validation
 - Developer-supplied expected system output on a common data set
 - Conformance tests on the Execution CLI interface components
 - Status checking code written by the developer to make sure processing progress is being made
- Simple parallelization model to process a test collection by dividing a data set into 'Chunks'
 - Each '**chunk**' is a set of videos/activities that a system will process on a computation Node
 - Each chunk must be processed separately
 - Systems can leverage content extraction/processing within chunk
 - The developer will implement a CLI component to efficiently design the chunks for their system
 - Could be: single video/single activity → single video/all activities → all videos collected at T_o /all activities
- Fault detection and recovery
 - Simplest user model possible - chunk processing either fails or succeeds
 - Compute performance on intermediate and incomplete test collection runs

Submission Information

- System Name
- System URL and access credentials
- Supported evaluation task: AD
- Known activities: the set of pre-trained activities
- Validation set output:
 - The file json, activity json, chunk json, and the output produce by the developer at the developer's site.
- Compute Node Limits:
 - Activities-per-chunk – max number of activities processable on a compute Node
 - File-streams-per-chunk – max number of camera views per compute Node

ActEV Evaluation CLI Overview

- NIST implemented
 - ActEV-get-system – Downloads a credentialed, web-accessible content into a NIST-supplied directory name <SYS>.
 - ActEV-validate-system – checks the structure of a <SYS> directory after *ActEV-system-setup* is run. Checks for expected API executables, required command line options, and existing system output for validation sets
 - ActEV-exec – a default wrapper script that calls a developer-implemented API in <SYS> given the system config file, file json, activity json, chunk json, video location, system cache dir. Captures time stamps, resource usage, etc.
 - Supports: Independent video/activities videos and activities are processed independently
 - ActEV-validate-execution - Test the execution of the system on each validation data set provided in <SYS> comparing the newly generated to the expected output and the reference
- Developer- implemented from common stub provided by NIST so as to have a consistent command line.
 - <SYS>/bin/ActEV-system-setup – runs any compilation/preparation steps for the system. <SYS> passes *ActEV-validate-system* after completion. Only this step should expect un-fettered internet access.
 - <SYS>/bin/ActEV-design-chunks* – given a file json and activity json, produce a chunk json that is suitable for the system
 - <SYS>/bin/ActEV-experiment-init* – specifies the system config file, file json, activity json, chunk json, video location, system cache dir, start servers (if used), starts cluster (future functionality)
 - For a given ChunkID:
 - <SYS>/bin/ActEV-pre-process-chunk* – specifies the ChunkID
 - <SYS>/bin/ActEV-process-chunk* – detection for ChunkID
 - <SYS>/bin/ActEV-post-process-chunk* – fusion within ChunkID
 - <SYS>/bin/ActEV-reset-chunk* – delete all cached information for ChunkID so that the chunk can be re-run
 - <SYS>/bin/ActEV-merge-chunks* – returns NIST-compliant, scorable system output for the listed chunks. Can be called at any time to get intermediate results, minimal computation expected
 - <SYS>/bin/ActEV-experiment-cleanup* – close any servers, terminates cluster (future functionality), etc.
 - <SYS>/bin/ActEV-status – executable at any time after *ActEV-experiment-init* exits and before *ActEV-cleanup* exits.
 - Report 'ok' or times out
 - Retrieves log files

* CLI components that must be implemented so that they will work within a confined network that does not have general WWW Access

ActEV Evaluation CLI Definitions: <SYS> directory

- <SYS> - a directory, created to contain all filesystem files for use by the analytic
 - For example:
 - NIST executes: ActEV-get-system -u <http://some.com/forJon> -S "/tmp/Test1"
 - To build a CLI Compliant <SYS> directory
 - /tmp/Test1/bin/ActEV-system-setup.py
 - /tmp/Test1/bin/...
 - /tmp/Test1/myDockerInfo/...

ActEV Evaluation CLI Compliant <SYS> directory

The <SYS> directory must be structured as follows:

../<SYS>/bin/ - The CLI-compliant executables (e.g., ActEV-validate-system.py, etc.)

../<SYS>/src/ - Developer-implemented entry points (See the CLI GIT Repo)

../<SYS>/container-output/ - a directory containing the site-generated output for at least one validation dataset. For each data set, the developer will provide the file, activity, chunk, and output jsons used/generated by the system on the developer's hardware. The naming convention will be:

<DATASET>_<FILE>.json

Where: <DATASET> ::= one of published data sets. E.g., ActEV1B-V1

<FILE> ::= "file", "activity", "chunk", "output"

ActEV Evaluation CLI Definitions: Chunks

- A 'chunk' is a set of video files and activities that the user can process independently on a DIVA Node.
- Chunks are specified in a new, 3rd input JSON file for the system

Example: Define 2 chunks with two videos each processing all activities

Activity JSON

```
{ "Closing": {  
  "objectTypes": [ "People" ],  
  "objectTypeMap": { "People": "*People*" },  
  "Closing_Trunk": {  
    "objectTypes": [ "Construction_Vehicle", "Vehicle" ],  
    "objectTypeMap": { "Vehicle": "*Vehicle*" },  
  "Entering": {  
    "objectTypes": [ "Construction_Vehicle", "Vehicle" ],  
    "objectTypeMap": { "Vehicle": "*Vehicle*" }  
  }  
}
```

File JSON

```
{ "VIRAT_S_000003.mp4": {  
  "framerate": 30.0, "selected": { "1": 1, "20941": 0 }  
},  
"VIRAT_S_000200_04_000937_001443.mp4": {  
  "framerate": 30.0, "selected": { "1": 1, "15165": 0 }  
},  
"VIRAT_S_000200_06_001693_001824.mp4": {  
  "framerate": 30.0, "selected": { "1": 1, "3908": 0 }  
},  
"VIRAT_S_000202_01_001334_001520.mp4": {  
  "framerate": 30.0, "selected": { "1": 1, "5566": 0 }  
}  
}
```

Chunk JSON

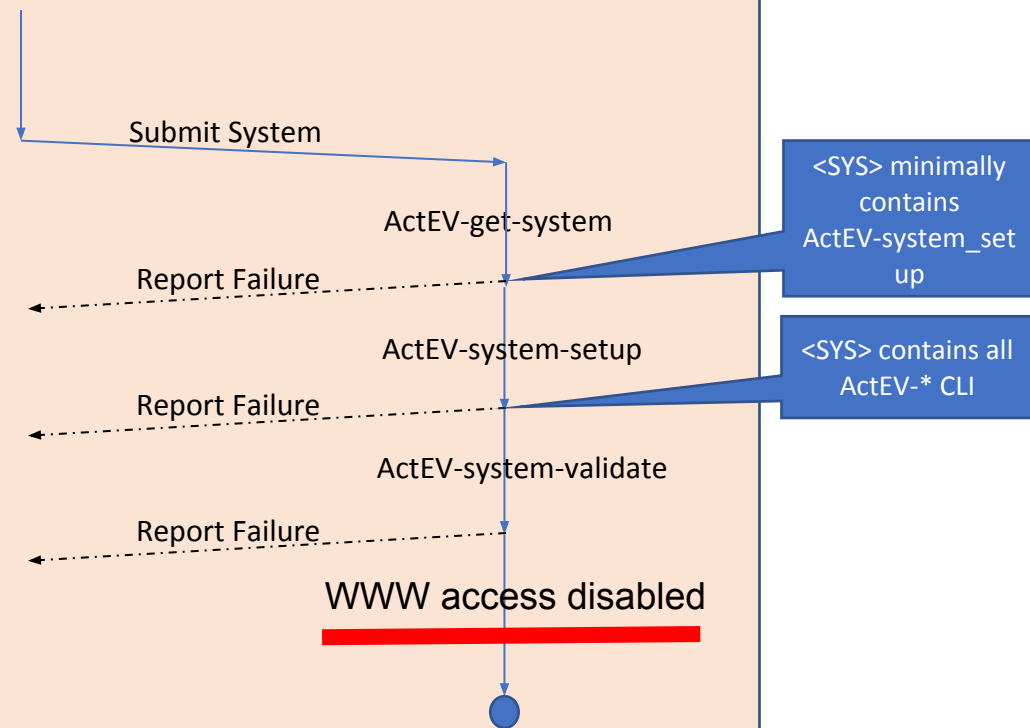
```
{ "Chunk1": {  
  "activities": { "Closing", "ClosingTrunk", "Entering" }  
  "files": { "VIRAT_S_000003.mp4",  
             "VIRAT_S_000200_04_000937_001443.mp4" }  
},  
"Chunk2": {  
  "activities": { "Closing", "ClosingTrunk", "Entering" }  
  "files": { "VIRAT_S_000200_06_001693_001824.mp4",  
             "VIRAT_S_000202_01_001334_001520.mp4" }  
},  
}
```

ActEV Evaluation CLI Process Diagrams

Prepare Phase

Developer

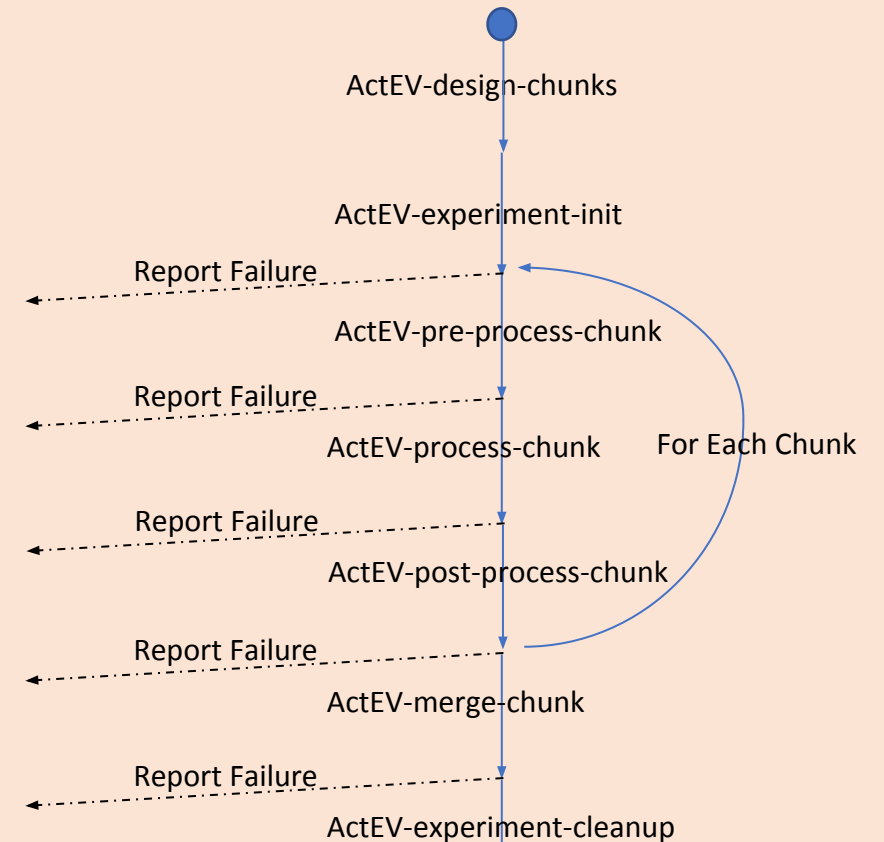
NIST



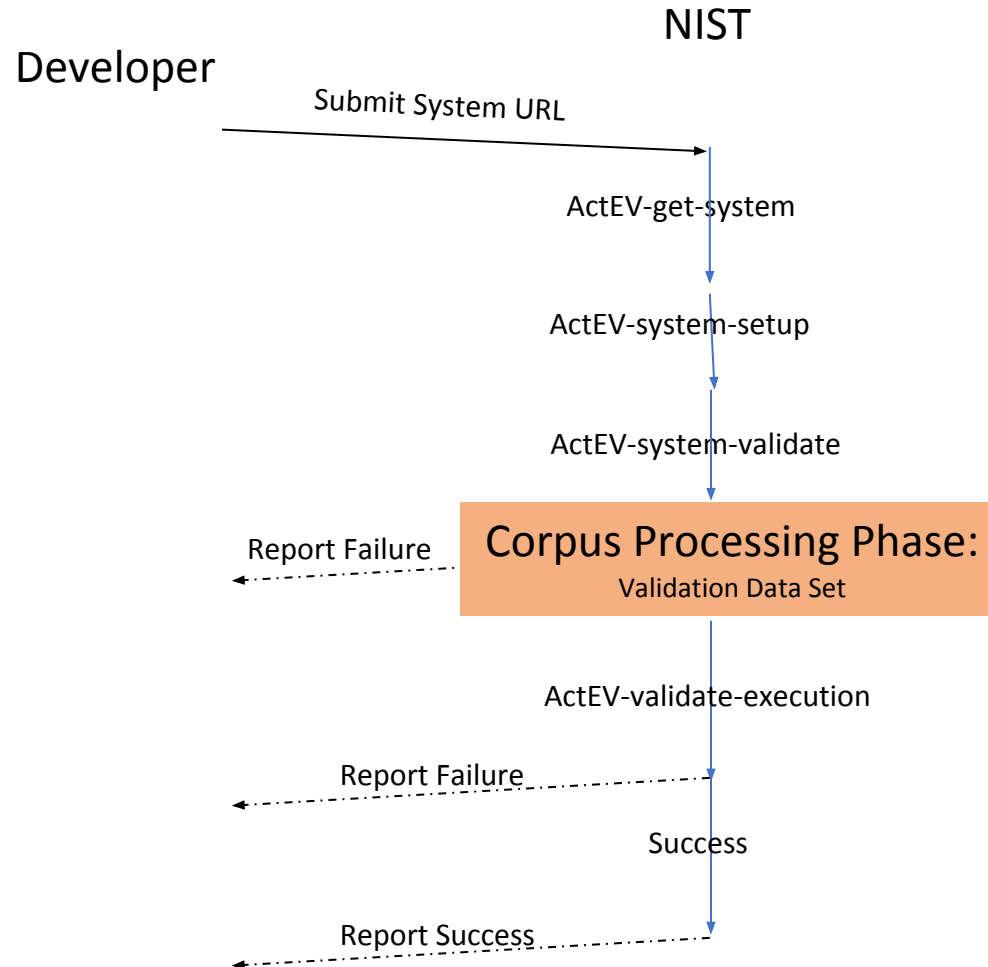
Corpus Processing Phase

Developer

NIST



System Delivery Validation Process



Slide Version History

- 20180907 - V1.1
- 20190225 - Added contextual slides. No content changes