APIs and CLIs for Independent Evaluations Version 1.3

April 16, 2021

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 Python 3

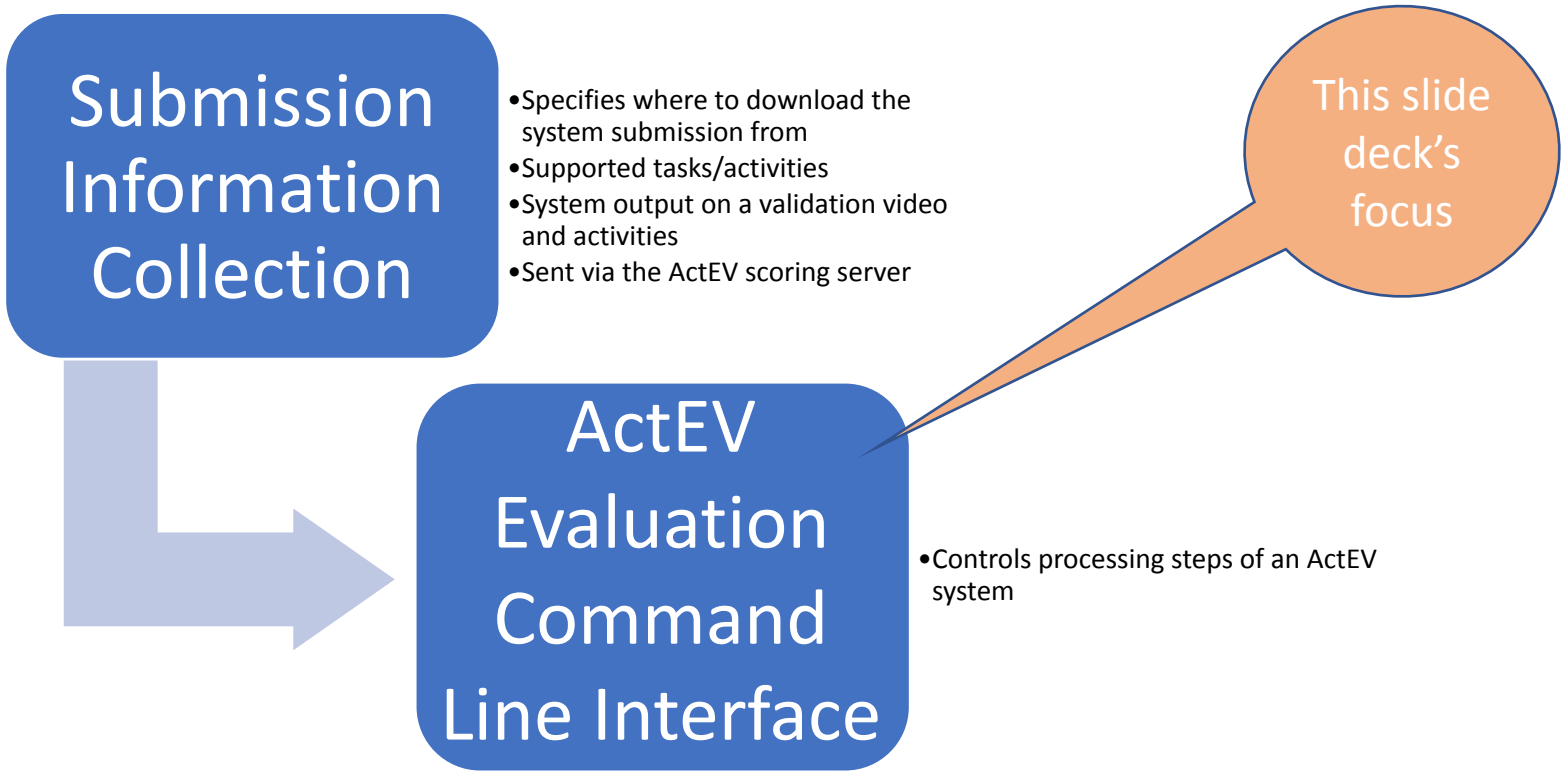
What is an “Abstract Command Line Interface”?

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” defintion 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
This slide deck's focus
ActEV Evaluation Command Line Interface
•Controls processing steps of an ActEV system

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_0/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
- Facility type
  - KF: Known Facility, 1080Ti GPUs, EO+IR data, Known Activities.
  - UF: Unknown Facility, 2080Ti GPUs, EO, Known and Surprise Activities.
- Validation set name output:
  - The file json, activity json, chunk json, and the output produce by the developer at the developer's site.
- Compute Node Limits:
  - File-streams-per-chunk – max number of camera views per compute node within a chunk.
- Produce Proposal Outputs:
  - The processing pipeline will use the CLI options to extract proposal outputs during execution.
- Produce Localization Outputs:
  - The processing pipeline will use the CLI options to extract localization outputs during execution.
- Computation Set Size
  - MicroSet has a smaller set of parts. FullSet can be triggered anytime later if MicroSet selected.

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/activites 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. Also test the JSON schema of the localization outputs.
- 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 unfettered internet access.
  - <SYS>/bin/actev-train-system* – an optional call where the system prepares to detect surprise activities (e.g., builds models). The system must write models and any other derived information into the <SYS> directory so that they can be used on subsequent executions of the system WITHOUT retraining.
  - <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). Also tells the system if localization / proposal outputs are enabled or not.
  - 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. Localization and proposal outputs are written during this step.
  - <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” }
  },
}
```

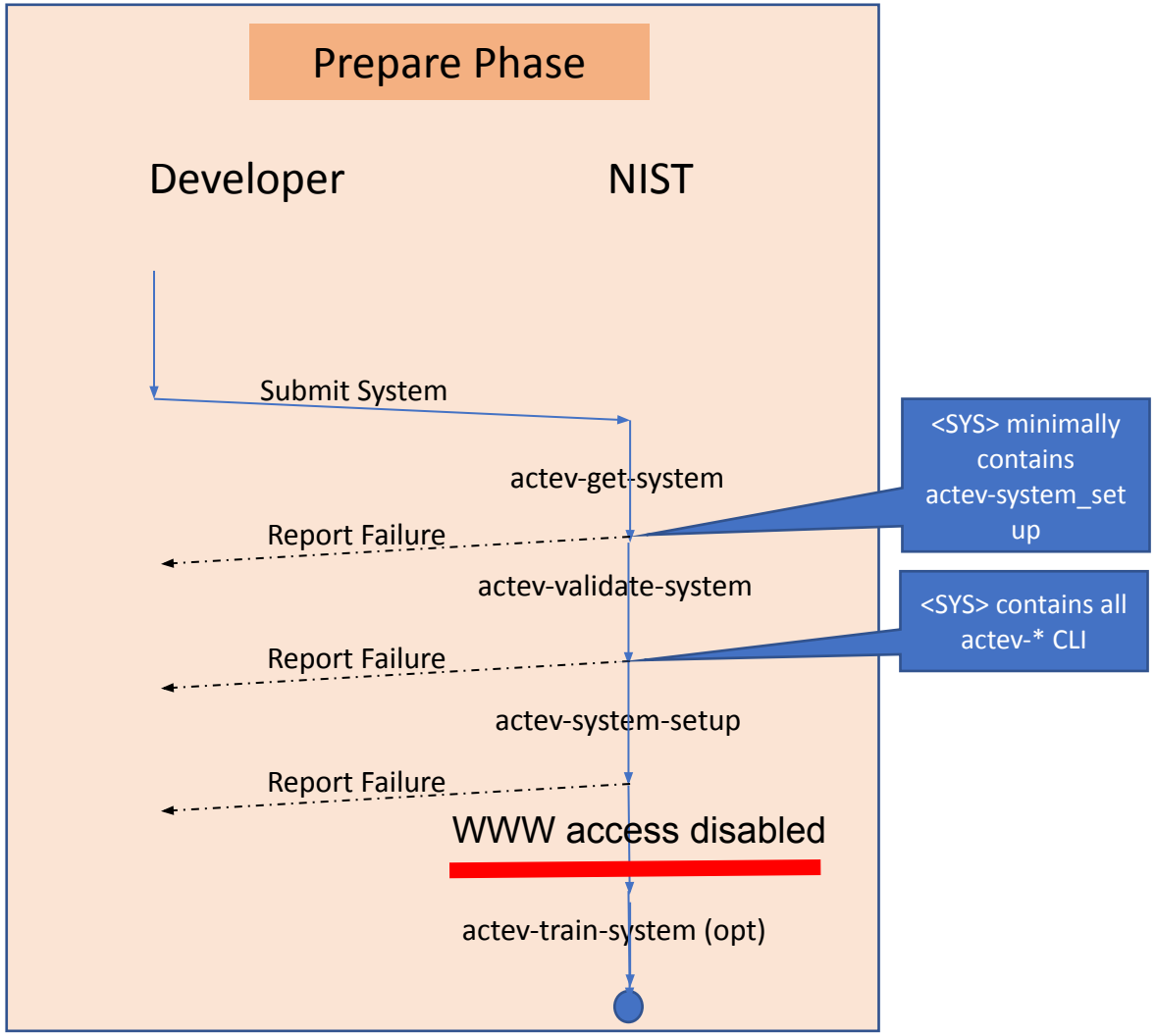
ActEV Train System - Surprise Activity Training

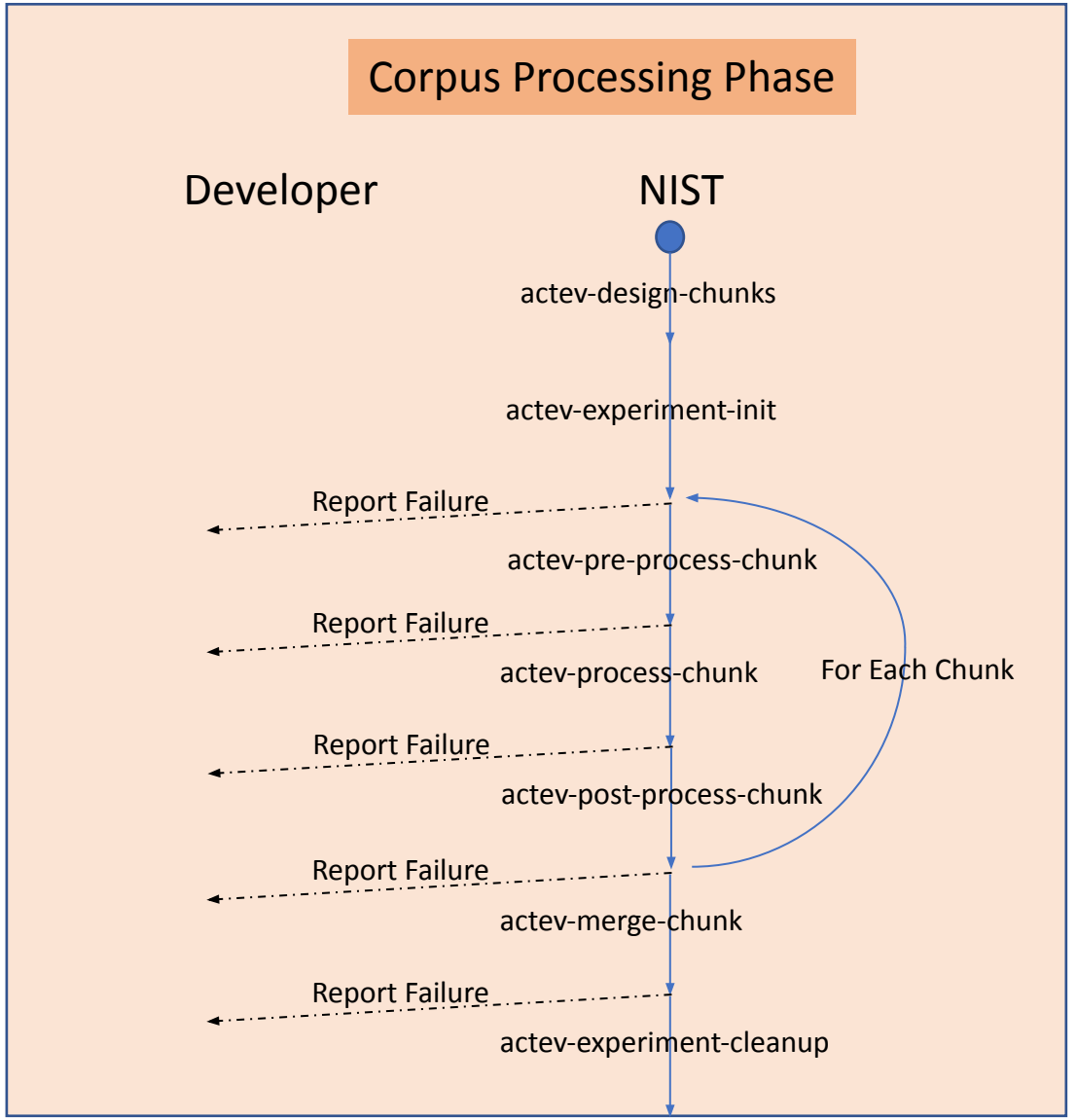
- Surprise Activities (activities defined for the system during testing) is a new feature of the ActEV CLI
- *actev-train-system* is the only new command. There are two inputs:
  - activity-index.json - The JSON file that defines the activities the system must be prepared to detect.
  - training-data-dir - A direct contain video and annotation content referred to defined in the activity-index.json.
- The command should:
  - Be called only once
  - Add any derived data concerning the activities to the system directory to support subsequent execution of the system

Notes on proposal and localization outputs

- proposal outputs
  - contains the proposals your system generated while processing the video. The CLI collects the produced file but the inspection of the output is a fully manual process. Please work with NIST to conduct this inspection.
- localization outputs:
  - same as a classic system output, but spatial localization objects are required. While this is similar to the ActEV AOD and AODT evaluation tasks, this output is provided to enhance information sent to down-stream processes.
  - the **presenceConf** value of localization objects is **optional**
  - schema is going to be validated during actev_validate_execution
  - scoring done by hand

ActEV Evaluation CLI Process Diagrams





System Delivery Validation Process With Surprise Activity Training