Data Collection Architecture Sub-Group

Charles Schmidt

September 29, 2020

MITRE | SOLVING PROBLEMS FOR A SAFER WORLD™

Objectives

- Support tool interoperability through standardized interfaces between SCAP collection roles

- Support expanded SCAP processes that require architectural support
  - Continuous monitoring
  - Expanded collection scope (cloud, IoT, etc.)

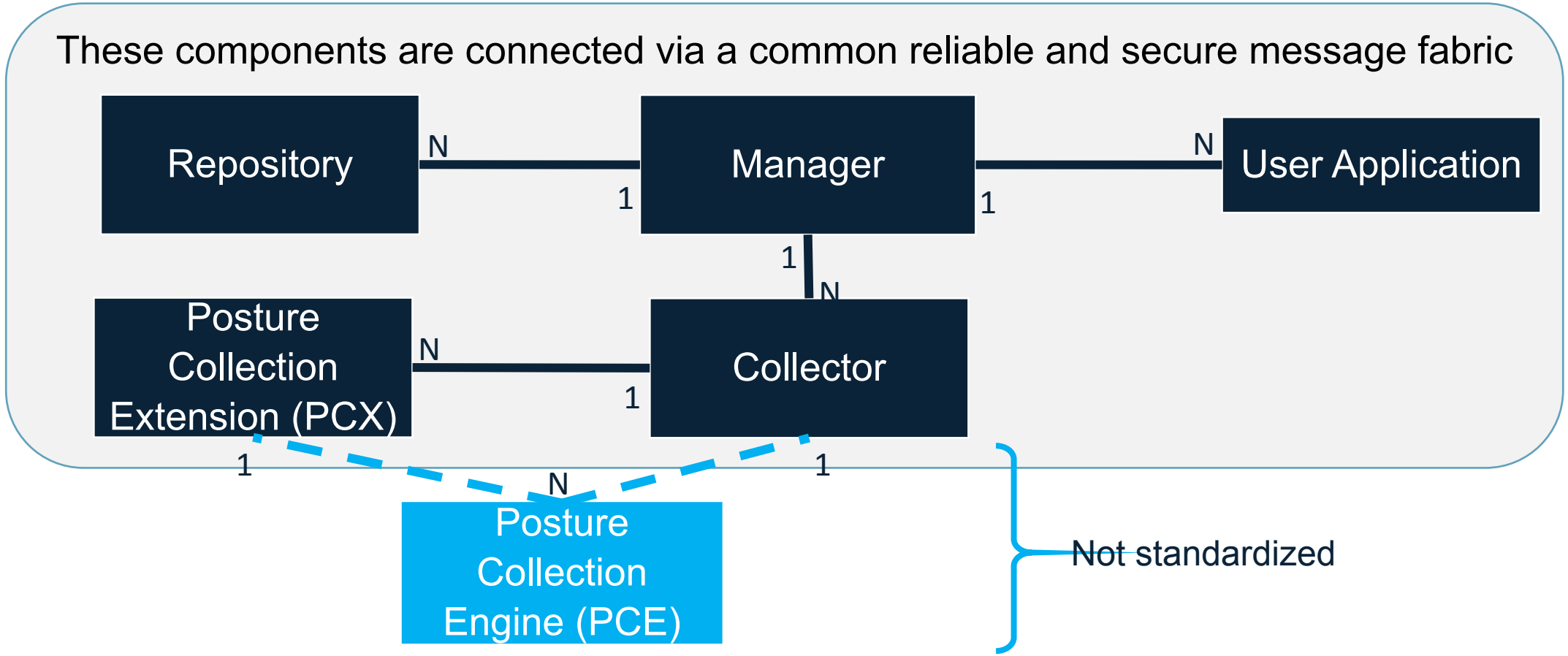
MITRE

SCAP v2 Logical Architecture – Key Ideas

- Support flexible implementation methods
  - Components can be combined or divided – key is that they allow standardized interactions at key points
- Support existing tools
  - Data collection agents are not standardized - accommodate existing tools
- Extensible
  - Allows dynamic addition of new collection capabilities
- Intelligent
  - Can take a collection request and figure out the targets and what tools to use

SCAP v2 Logical Architecture

- A framework of key components, loosely connected via standardized interfaces, for gathering information about enterprise assets



SCAP v2 Logical Architecture – Component Overview

- User Application – Where requests for information about enterprise state originate
- Repository – Storage for prior collection results, enterprise assets (a.k.a. collection targets), and other persistent control data
- Collector – Manages data collection for a specific set of targets – tightly bound to a set of PCEs
- Posture Collection Engine (PCE) – Handles actual data gathering. Not standardized but linked to a Collector or PCX that handles interfacing.
- Posture Collection Extension (PCX) – A “sub-Collector” connected via a standard interface. Allows new collection capabilities to be added to Collectors.
- Manager – Handles orchestration of components to perform collections

Key Functions of the Architecture

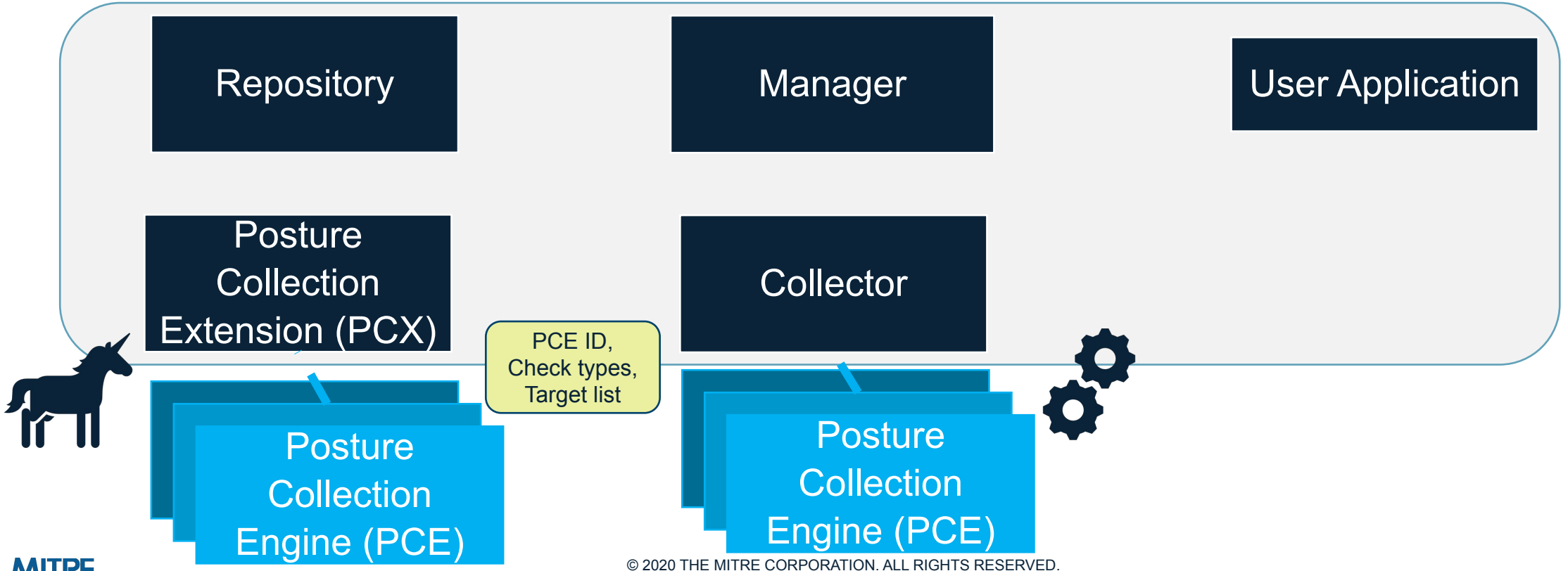
- Identify collection targets
  - Collectors gather and report potential collection targets to Repository
  - Manager matches collection targeting and applicability to Repository data
- Identify and task collection tools
  - Manager determines which Collectors to task for a given target set
  - Collectors map instructions to PCE/PCX capable of handling that type of collection
- Orchestrating continuous monitoring
  - Collectors set up and manage event-based/periodic checking as dictated
  - Repository monitored for new applicable targets to update Collectors

A Walkthrough – Point-in-time Collection

- The following slides walk through how the architecture handles point-in-time collection
- For clarity, the walk-through assumes different software applications in each role
- Some details remain TBD
  - E.g., Communications paths are under experimentation – walkthrough just notes data source and sink without specifying path or mechanism
- Please provide feedback! Nothing is final, but this is what we are thinking…

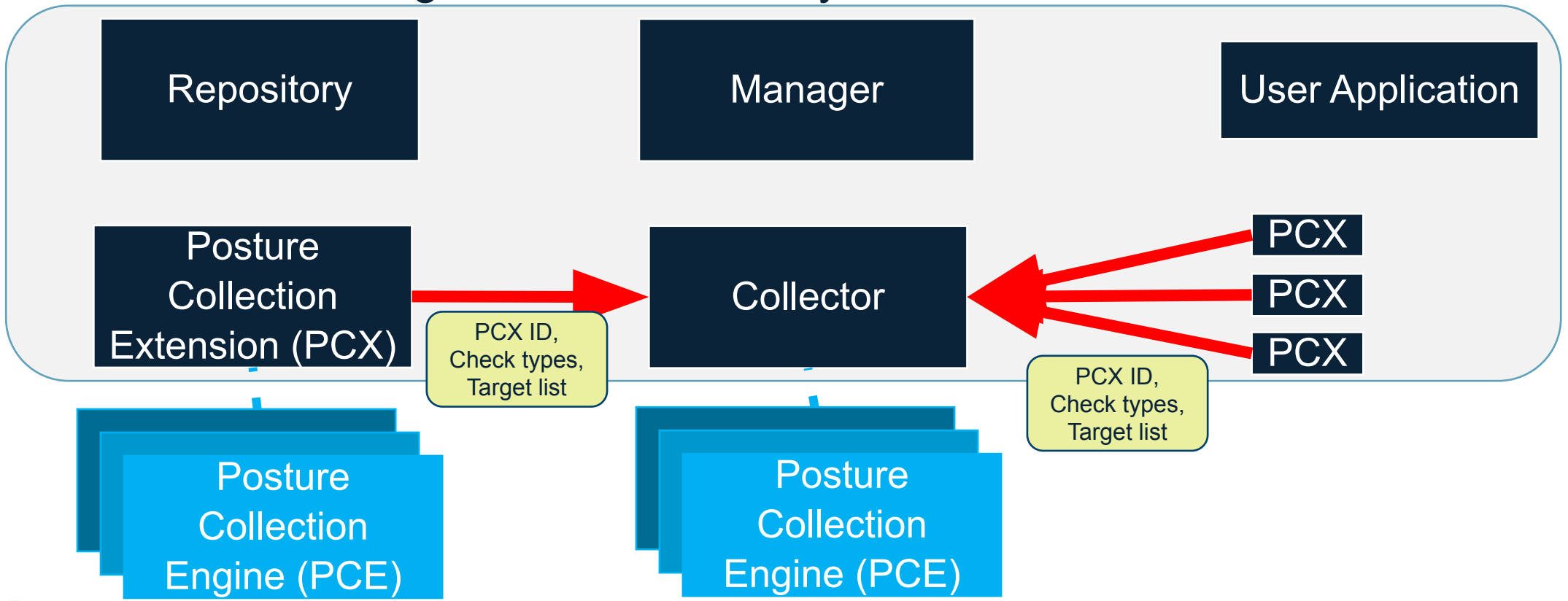
Walkthrough – Setup (PCE and Target identification)

- PCEs are connected to Collectors/PCXs
  - Process not standardized – the Collector/PCX just needs to be aware of active PCEs it can task – Collector/PCX and its PCEs may be from same vendor
  - Includes identification of collection targets PCEs can act on and supported checks



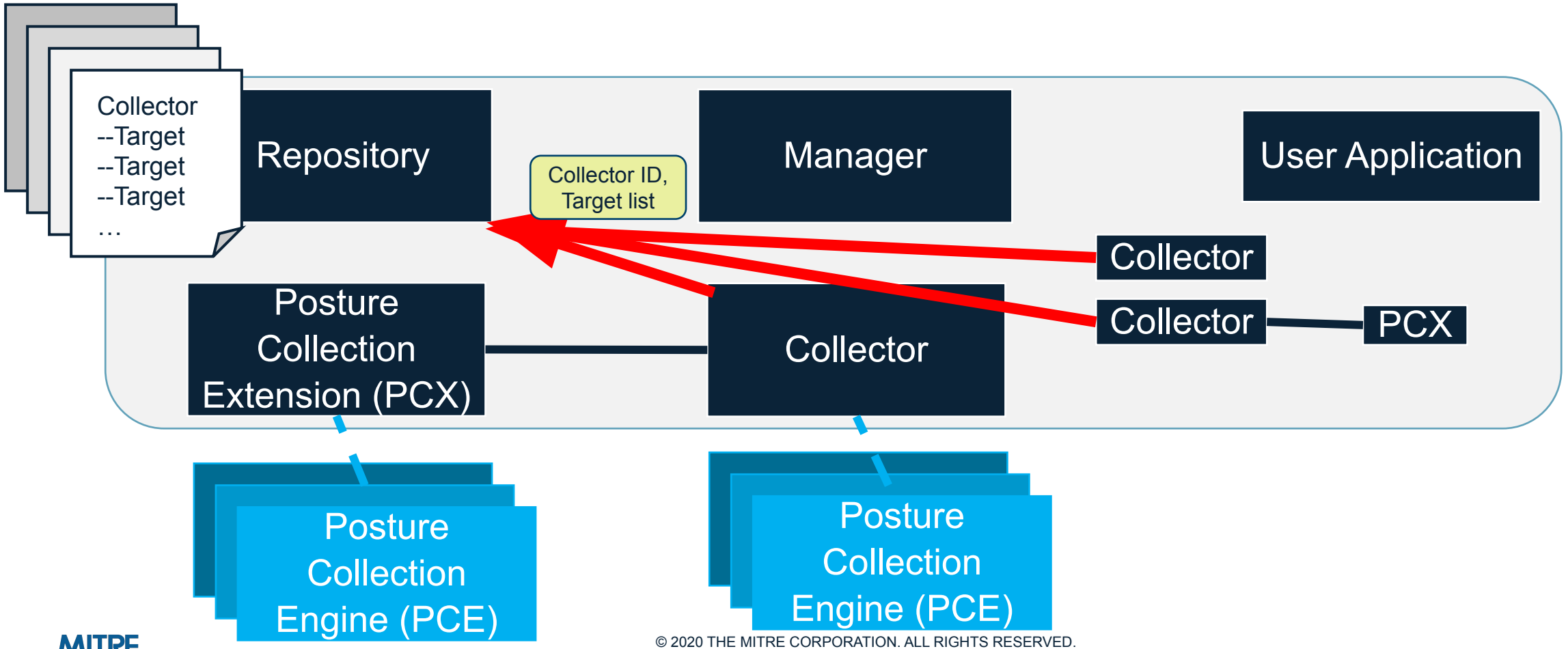
Walkthrough – Setup (PCX Registration)

- PCXs register themselves to a Collector
  - Identify the types of checks they are capable of processing
  - Include list of targets from which they can collect



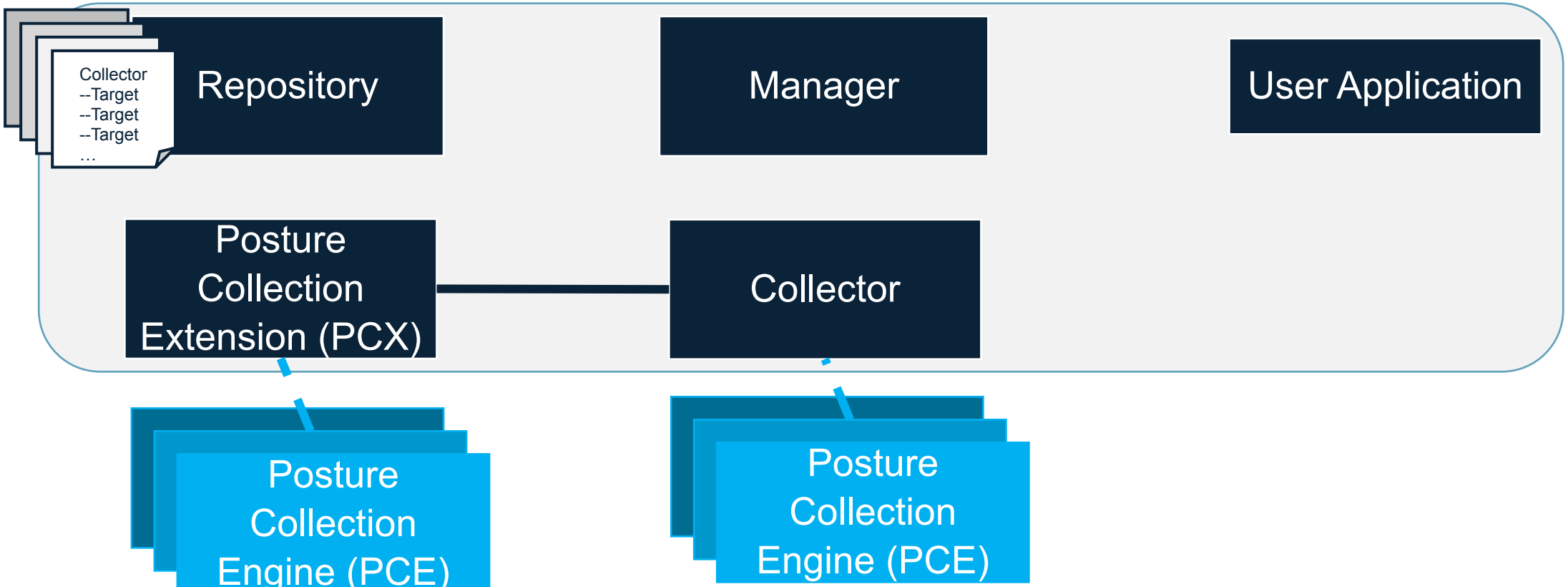
Walkthrough – Setup (Collector and Target Registration)

- Collectors store combined (Collector + all its PCXs) target set in the Repository



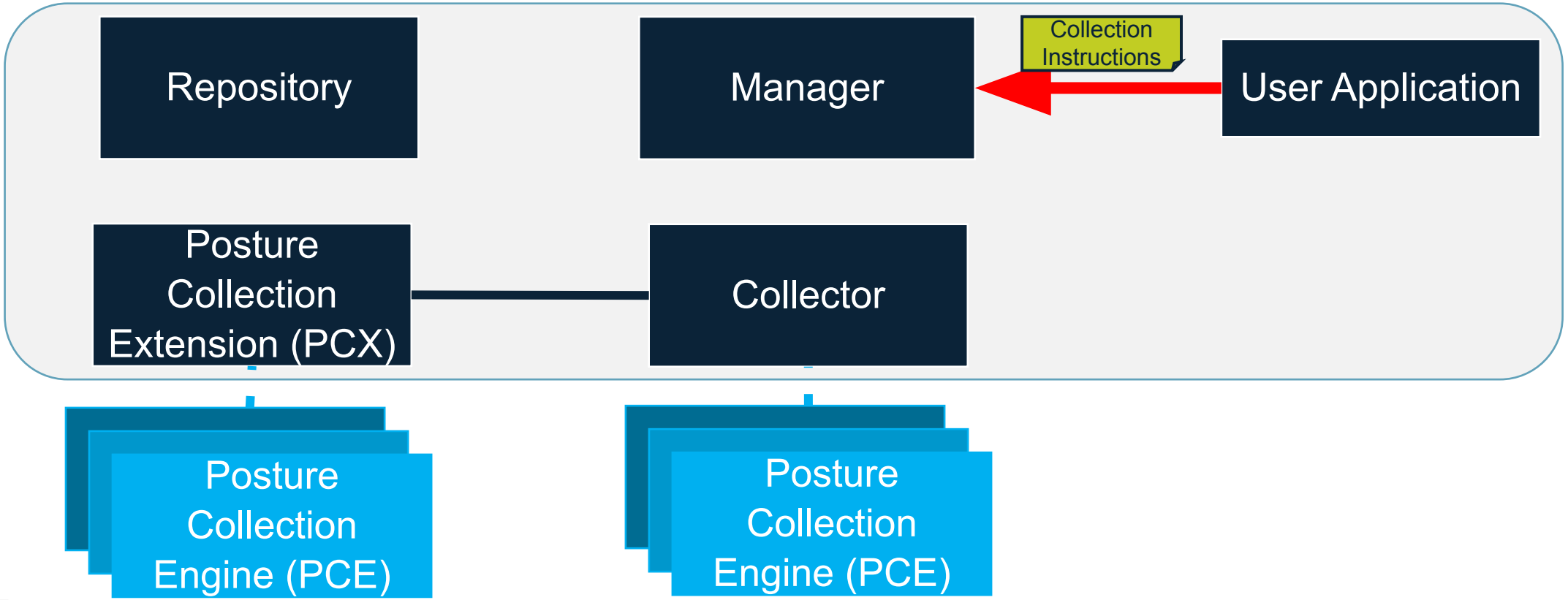
Walkthrough – Setup (Upon Completion)

- Collectors and PCXs are aware of all their PCEs, the check types they support, and the targets those PCEs collect from
- Collectors know all their available PCXs, the types of checks they run, and their targets
- The Repository has a list of all active Collectors and a (total) target list for each Collector



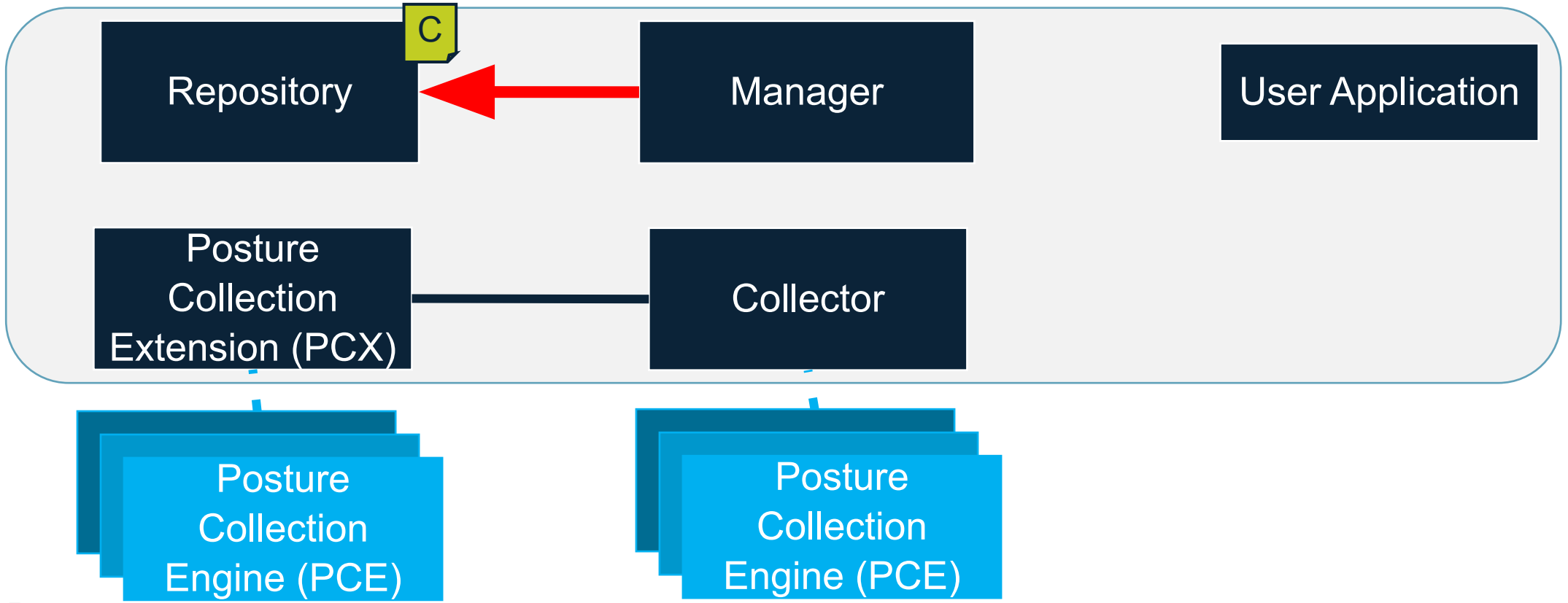
Walkthrough – Collection (Initial Tasking)

- User Application sends collection request to manager
  - Includes collection instructions, freshness guidelines, non-technical targeting (e.g., organizations), method guidance, and return format



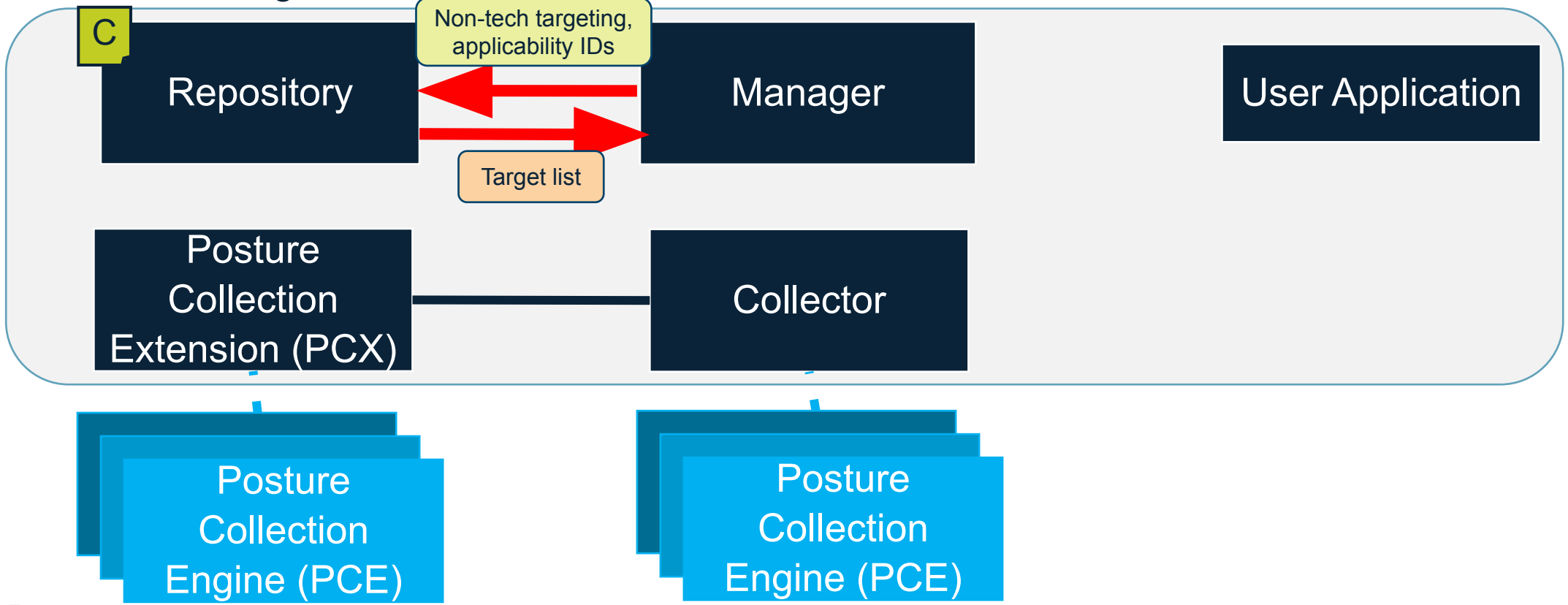
Walkthrough – Collection (Store Tasking)

- Manager stores the collection request and parameters in the Repository



Walkthrough – Collection (Targeting and Applicability)

- Manager queries the Repository for the intersection of
  - Targets within any non-technical targeting criteria
  - Targets that match applicability checking (see next slide)
- Result is the target set of the collection



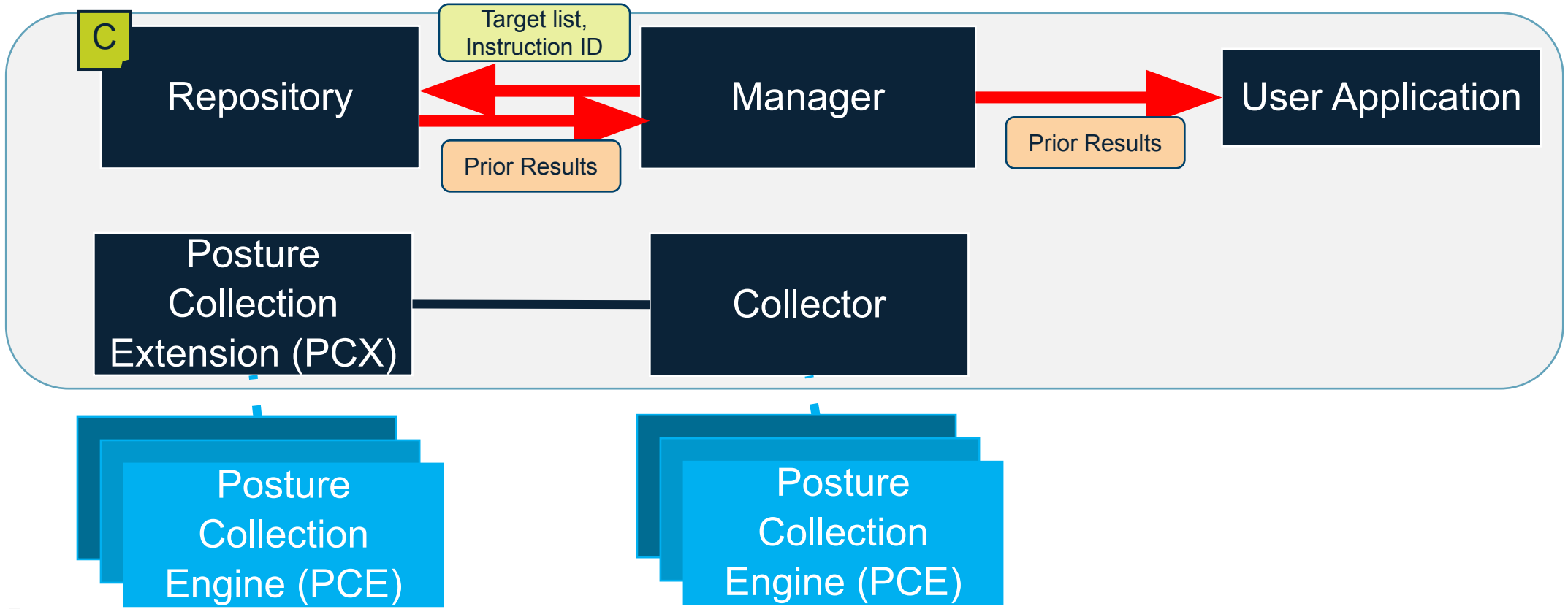
How Pre-Collection Applicability Checking Works

- All checks, including applicability checks, have unique IDs
- All collected data within the framework, including results of running applicability checks, are stored in the Repository (unless instructions say otherwise)
  - Collected data is linked to both the target and the check ID
- Using the Repository to do pre-collection applicability checking:
  - Given an applicability check's ID, add all targets for which there are no associated results for that check ID
  - Add all targets for which an existing result indicates anything other that confirmation of inapplicability (i.e., positive applicability and some ERRORs)
  - Resulting list is all targets that are not known to be inapplicable
- One might post a set of applicability checks on start-up to pre-populate applicability data. Otherwise, these checks occur as part of other collections.

MITRE

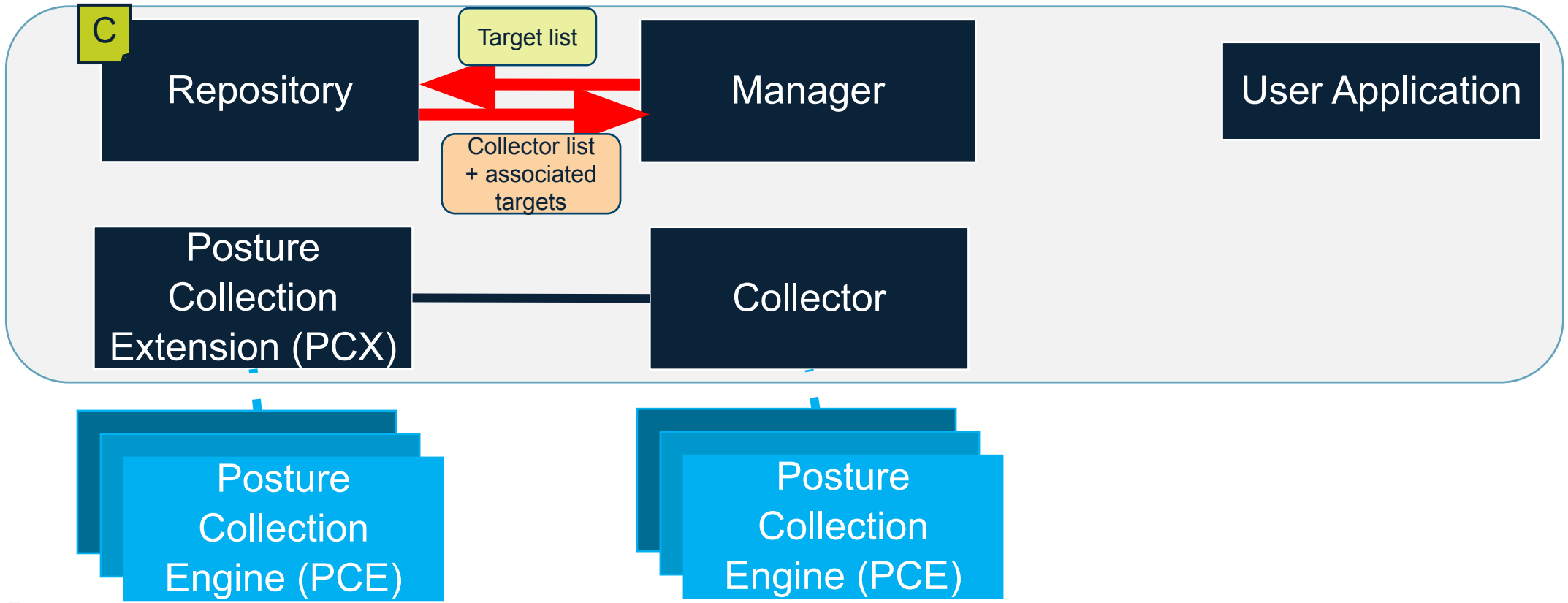
Walkthrough – Collection (Find Prior Results)

- Manager queries the Repository for prior results
- Repository looks for targets with suitably fresh results for specified collection instruction ID and using specified collection methods



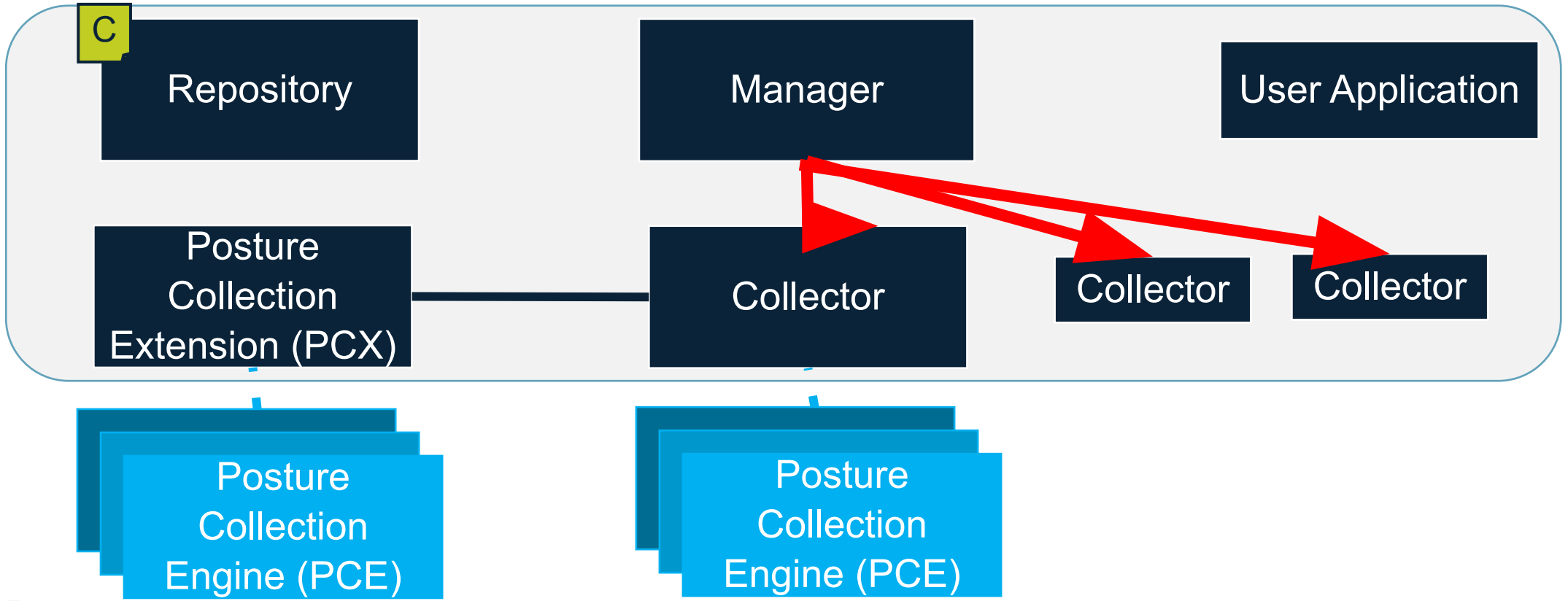
Walkthrough – Collection (Collector Identification)

- Query Repository by sending the target list (remove targets with prior results)
- Repository returns IDs for all Collectors covering at least one target



Walkthrough – Collection (Collector Tasking)

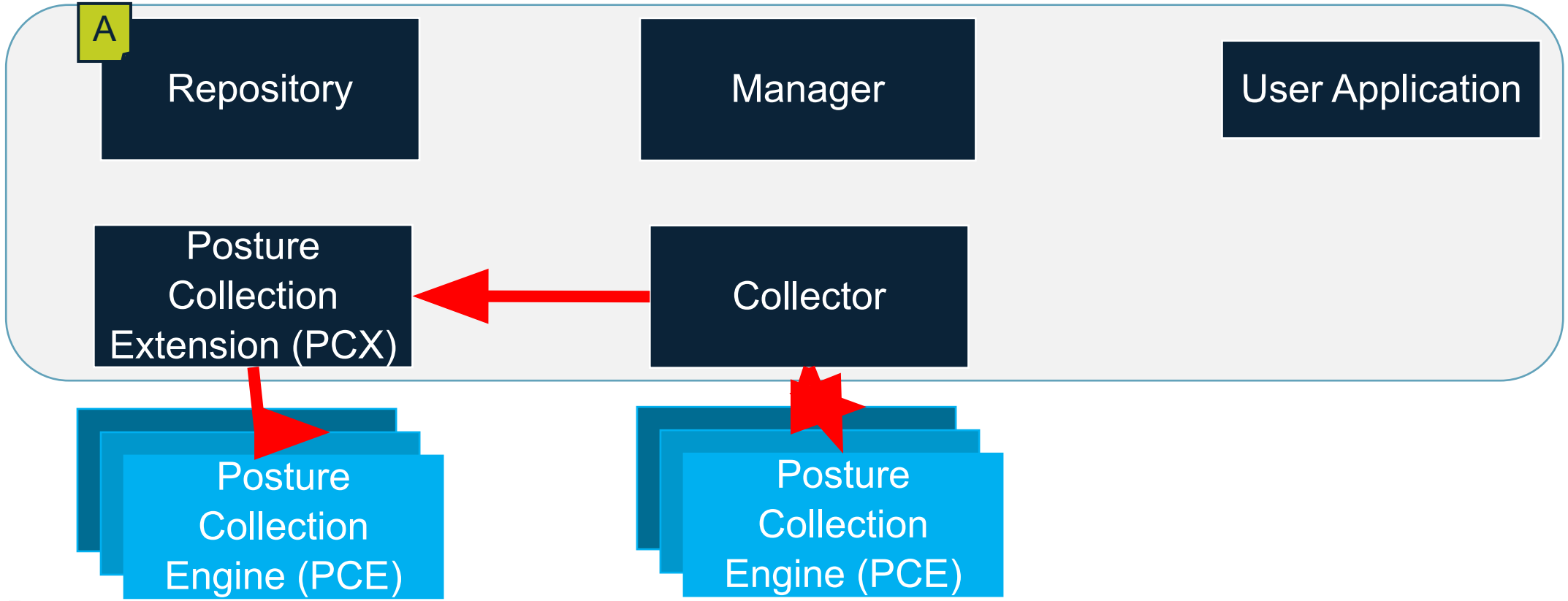
- Manager tasks each Collector
  - Each Collector is given explicit target lists to avoid duplicate collection



MITRE

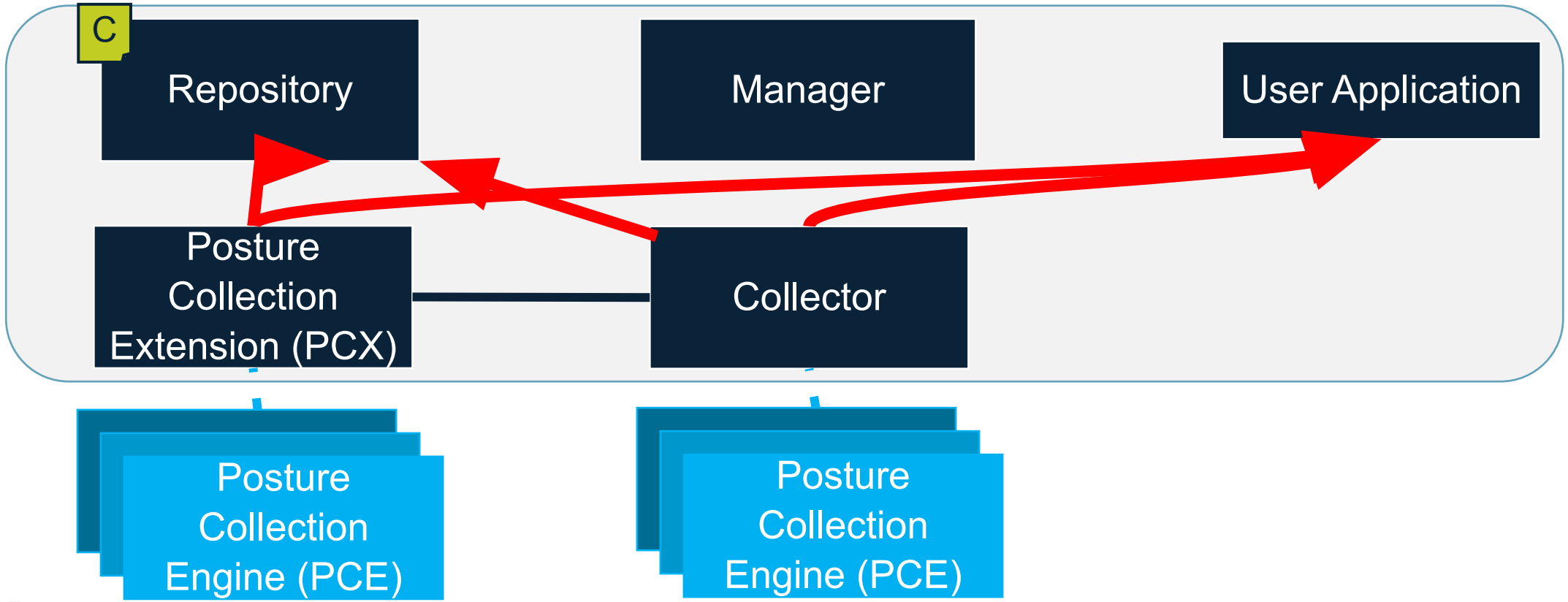
Walkthrough – Collection (PCE/PCX Tasking)

- Collector examines type of each check in collection instructions
- Collector examines constraints on data collection methods (e.g., ATO constraints)
- Parcels out checks to its PCEs/PCXs based on ability/fitness to collect
  - Tasking of PCEs may involve instruction translation and proprietary APIs



Walkthrough – Collection (Publish/Store Results)

- Collection results annotated/normalized by Collector
- Results added to Repository and User Application is alerted to results



Continuous Monitoring – Quick Walkthrough

1. Collect baseline for monitoring – mostly same as point-in-time collection
2. Collectors set up periodic or event-based checks per guidance in collection instructions
3. Filter periodic check findings based on their “significance”
   - “Significant” if a changed measurement changes a check result
4. Publish significant measurements
5. Potentially update target list as new targets discovered/registered in Repo.
6. Continue until User Application ends the monitoring
   - Delete periodic and event-based checks when monitoring ends

Important Ignorance Invariants

- The Manager never needs to know the checking capabilities of the system
  - Manager tasks Collectors based on targets; Collectors manage capabilities
- Collectors never need to know about other Collectors
  - Each Collector operates independently
- The User Application doesn't need to know what enterprise assets exist or what type of assets there are
  - UA sends instructions and characterizes targets; Manager + Repository turns this into an actual target list
- PCEs never need to know they are part of the SCAP architecture
  - Collectors/PCXs handle all interfacing with PCEs

Technical Details to Work Out

- Continuous monitoring control? I.e., how does User Application specify which checks have what re-checking periods?
- User Applications need some control over the format of the results they receive
  - How are appropriate SCAP formats identified?
  - Do other filters get applied? (E.g., “Don’t return NOT APPLICABLE results”)
  - Should the format of data sent to the Repository be controlled separately?
- Language (and capabilities) of non-technical targeting for collection
- For applicability checking, does Manager extract checks from content or are applicability checks sent separately?

Open Questions

- Distribute results to User Application piecemeal or when complete?
  - Continuous Monitoring requires piecemeal, but what about baseline/point-in-time?
  - User Application gets results from Collectors/PCXs? From Repository? From Manager? From queues in the message fabric?
- What are acceptable languages for collection results?
  - Collectors will be responsible for converting PCE results to these formats
- Do we constrain collection instruction formats?
  - A PCE might take anything; Collectors ensure interoperability with its PCEs and are probably statically assigned to a given set of PCEs
  - No Constraint = better support for native instructions but means different Collectors will accept different input instruction formats, hampering interoperability
- During continuous monitoring, what are limits for determining if change is “significant”?
- Is it a problem for a single target to be in scope of multiple Collectors?

SCAP Architecture High-Level Specification

- Specification captures most of the points discussed in this presentation
  - Latest version: Aug 18, 2020 https://groups.google.com/a/list.nist.gov/forum/#!topic/scap-dev-endpoint/Uqdzr_ORqjM

- High level and explicitly notes many open questions
  - I.e., this isn't a specification to which one could implement

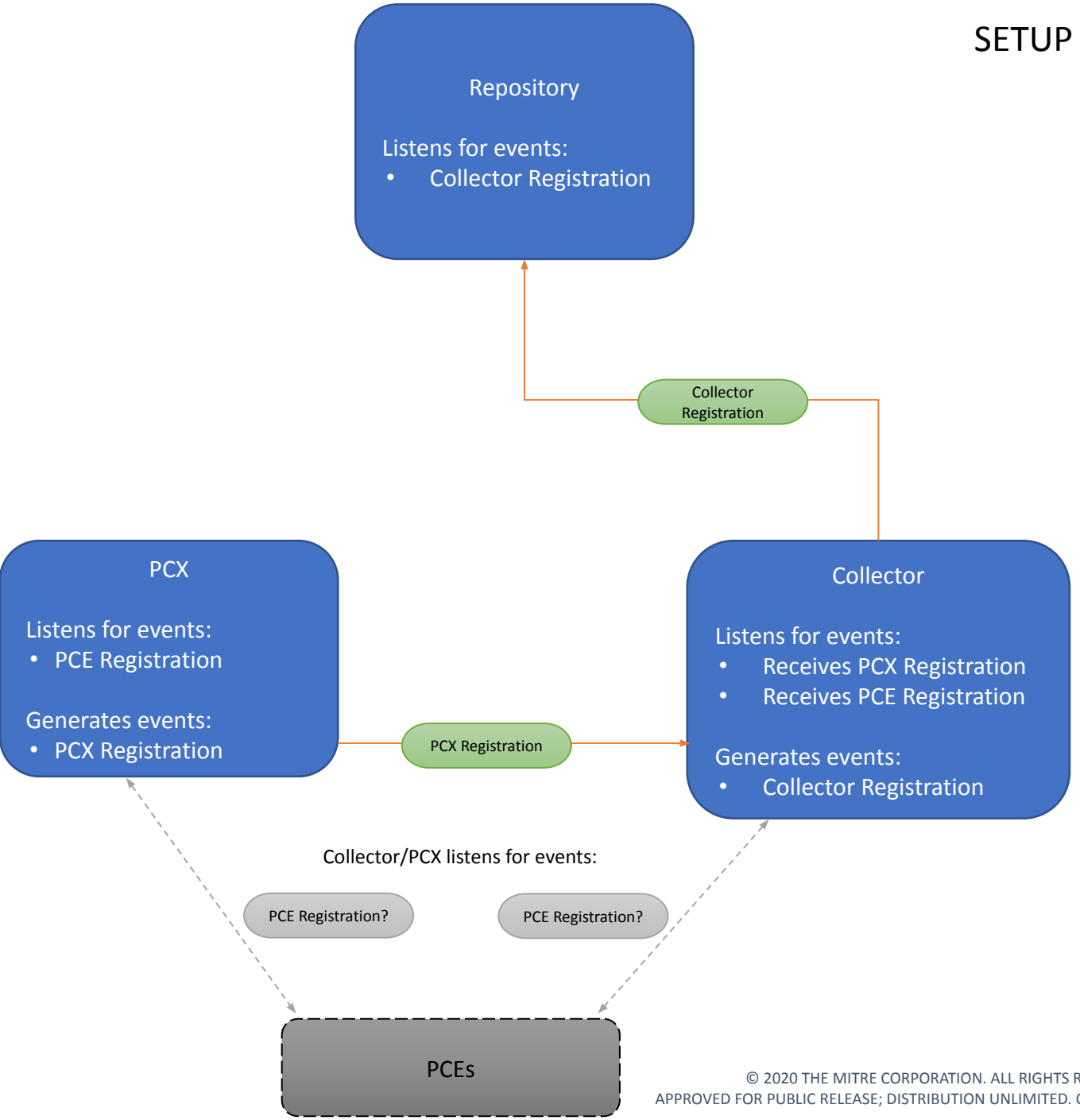
The SCAP Architecture Prototype

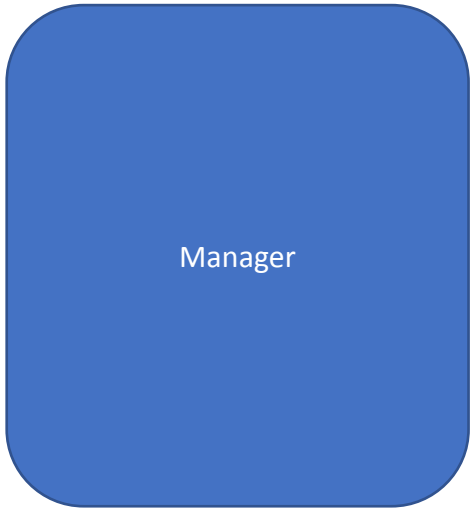
- Goal is to perform experimentation to help answer open questions and test technical solutions

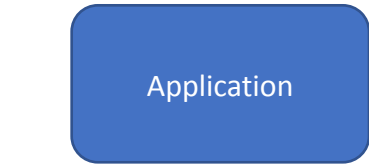
- Current prototype is written in Python and uses the OpenDXL message fabric
  - Neither of these reflect normative decisions

- Currently being developed at MITRE; shifting to the Open Cybersecurity Alliance shortly (will talk about that momentarily)

SETUP







Communication Type

- Query-response
- Event-based
- Proprietary

Red Fields are used by Collectors/PCX
Blue fields are used by the Manager
Remaining fields are for debug and maintenance

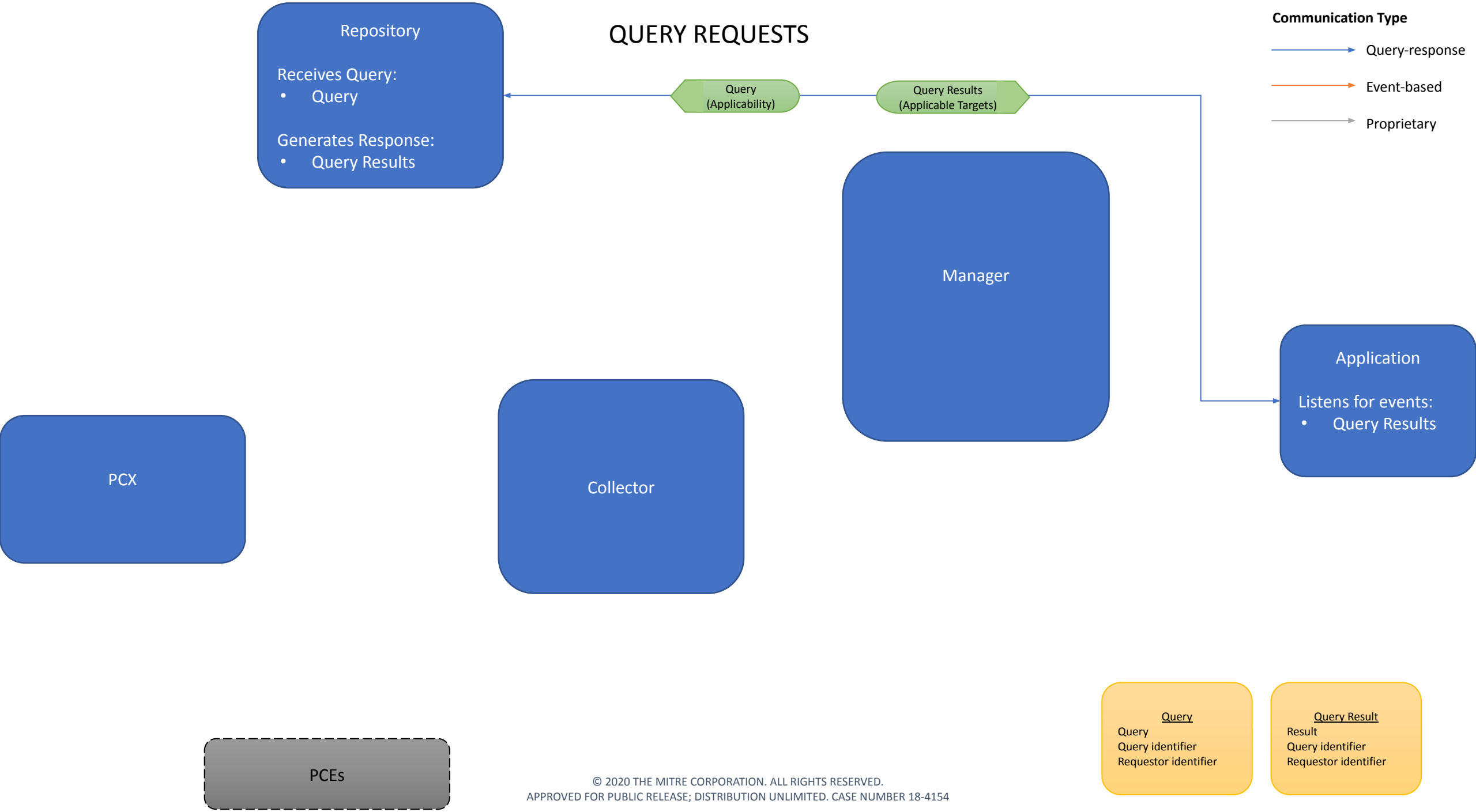
NOT STANDARDIZED
(Might not even come from PCE. This is just the information each Collector/PCX needs to know about its PCEs)

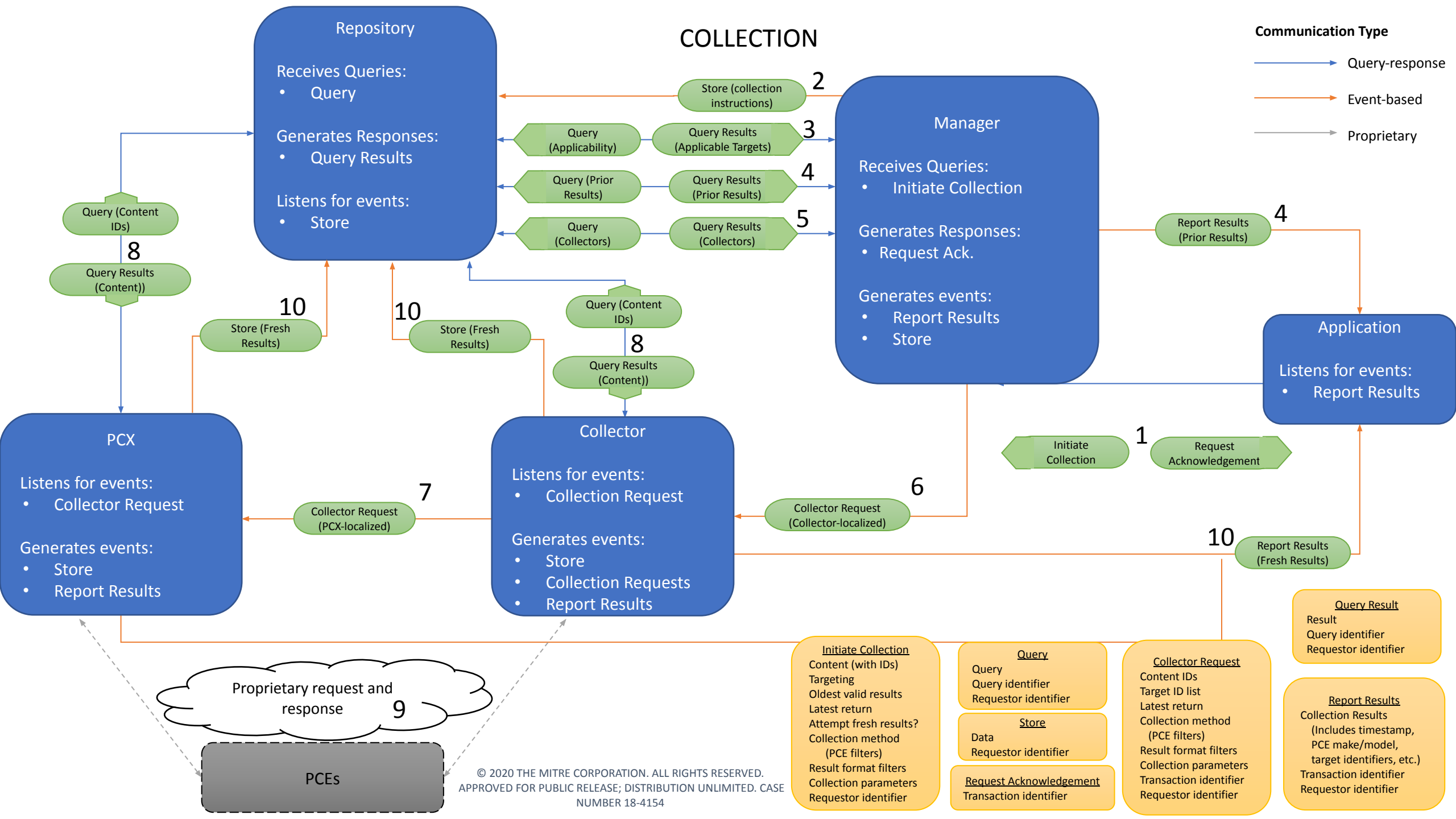
PCE Registration ?
- **PCE info list**
  - **PCE Identifier**
  - **Make & Model**
  - **Supported check types**
- **Target info list**
  - **Asset information (IP, MAC, Hostname)**

PCX Registration
- PCX info list
  - **PCX Identifier**
  - Make & Model
- **PCE info list**
  - **PCE Identifier**
  - **Make & Model**
  - **Supported check types**
- **Target info list**
  - **Asset information (IP, MAC, Hostname)**
- **Target identifier**

Collector Registration
- **Collector identifier**
- Collector Make & Model
- PCX info list
  - PCX Identifier
  - Make & Model
- PCE info list
  - PCE Identifier
  - Make & Model
  - Supported check types
- **Target info list**
  - **Asset information (IP, MAC, Hostname)**
- **Target identifier**

QUERY REQUESTS





Demo

MITRE

The Open Cybersecurity Alliance (OCA)

- "The OCA supports commonly developed code and tooling and the use of mutually agreed upon technologies, data standards, and procedures."

- OCA is an industry consortia supporting open standards and tools for cybersecurity
  - An OASIS Open Project

- One thing OCA does is support open source development projects that will enhance cybersecurity practices
  - The SCAP Architecture prototype was the first such project proposed from outside OCA

MITRE

SCAP Architecture OCA Project

- OCA will provide a GitHub repository, message board, and archiving support
  - MITRE/NSA are currently finalizing release of the code
- Many OCA members are interested in this work – gives us more eyes on the code and hopefully more participants in SCAP
- Some OCA members are vendors whose products align with PCEs – may give us opportunities to test the architecture with real vendor products
- Final architecture can serve as a reference implementation – OCA will host

MITRE

SCAP Architecture OCA Project (2)

- OCA requires that code development be public and archived
- OCA requires that the code product be released under suitable license (we are using Apache v2)

- OCA and its members will not dictate code design decisions
  - Architecture design will continue to come from the SCAP Data Collection sub-team
- Anyone can contribute – you don't need to be an OCA member

- Does not change management of SCAP in general – still managed by NIST and the SCAP community

In the Next 6 Months

- Continue development of the SCAP architecture prototype
  - NSA will lead development; hopefully others will help

- Capture lessons learned and design decisions from prototype in technical documents
  - Eventually, these will form the basis of a formal standard