**SERVICE DISCOVERY AND CONTROL**

 SUBMITTED BY WG-23: Artificial Intelligence and Application Hosting

**Introduction/Scope**

This work item proposes a mechanism for discovering and managing processing services. With the rise of artificial intelligence, containerized processing, service-oriented architecture, and microservices, there is a proliferation of processing services in the medical imaging space. Systems that use those services need to discover what services are available and to launch and control those services, for example, signalling a service to change its operational state (“start”, “stop”, “prepare for processing”, etc.).

A draft WG-23 white paper “AI Service Discovery and Control” and a preliminary outline on Diagnostic Radiology Applications – Hosting, Applications, and Manifests would inform the work to be done.

## **Limitations of Current Standard**

SERVICE DISCOVERY

● Part 19 Chapter 7 (Hosted Application Life Cycle) presumes the application was previously installed and configured on the host.

* Dynamic registration or discovery of hosted application capabilities is needed for load balancing, multiple processing services with similar capabilities, and to be able to instantiate applications prior to their use.
* Application Registration needs to inform hosts about the characteristics of a given processing service, and how to locate, instantiate or launch specific instances of a given processing service.
* Applications may also need to discover other Applications whose functions they can use.

● Part 15 Chapter H (Application Configuration Management profiles) uses LDAP and DNS. These are unsuitable for distributed systems.

* LDAP uses a root and relative hierarchical naming system for objects such as processing services. Every object name is fully unique within the full​ hierarchy. With externally hosted services, serving multiple sites, different and unique processing services could have the same name.
* In distributed WAN configurations (e.g., discovery of services via DNS where a client’s DNS system is defined by IT departments, or across many different DNS platforms) this can become problematic as externally hosted services may not be able to gather resources required as they can be behind firewalls. Although DNS is used to map the base URI components to IP addresses, what is described in DNS may not be the processing services themselves but for example, a manifest describing how to instantiate the service. URIs for resources can change, for example a request executed internally may use one host, while a request executed externally the resource must be accessed through a different host. Resources can also be moved and no longer discoverable at their initial URI. For processing services, a start URI, where an Application Manifest can be found is an approach that would allow a service provider to relocate an Application or apply load balancing. Exposing departmental or institutional DNS to the internet would be a security risk. There is complexity beyond DNS that this work will need to address.

SERVICE CONTROL

● Part 19 Chapter 1 (Application Hosting) observes in Chapter 1 that the API does not prevent remote execution, but acknowledges that it was designed for local execution

* The states in Chapter 7 (Idle, In Progress, Suspended, Completed, Canceled, Exit) do not allow for foreign microservices which rely on health and readiness.
* It considers ‘plugins’ not standalone discoverable services. It states Hosted Applications written to the standardized API would be able to run on any Hosting System that​ implements that standardized API. This needs to allow for RESTful services and be flexible to allow for new technologies which are host agnostic.

● Chapter 7.1 (Initialization) does not explicitly cover how applications are initialization, for example it does not cover instantiation of containerized workloads. It assumed that the Hosting System initialized a Hosted Application by issuing a run command or its equivalent with command line parameters to specify the end point references to be used for the interface.

● Chapter 8 is SOAP and XML-based; RESTful APIs and services are needed.

● Chapter 10 describes DICOM SOP Instances in XML; JSON is needed.

DATA EXCHANGE

● Part 15 Chapter H does not cover how applications outside an organization's firewall address data from internal exposed resources or how applications inside the firewall access data when it is transient between systems (data mobility). The LDAP and DNS limitations described for service discovery also apply for network addressed data and resources used by services.

## **Description of Proposal**

● Introduce an Application and it’s Manifest that describes the Application to a Platform/Host (unique identifier, what does the application does, what is needed to run the application and/or where it is located, what are the inputs and outputs, what are the connectivity endpoints, what transforms and data preparation are necessary). A Manifest will be used to describe DICOM services or Applications in general, that is “I provide services X, Y, Z”, and the ability to apply those services to specific instances, so that both discovery of what services are available can be found and what services can be provided.

● Review the suitability of the Open Application Model (<https://oam.dev/>) for DICOM service discovery and control, and specifically application manifests. Briefly, in OAM:

○ A developer creates an application or service. To deliver it to users, the developer defines how to discover, instantiate and interact with it in a YAML manifest. This manifest encapsulates a workload and the information needed to run it.

○ An applicationoperator deploys instances of an application, and configures it with “operational traits” (parameters that control things like instance replication).

○ The application developer and application operator have so far described an application and its operational characteristics in platform-neutral terms. The power of the Open Application Model comes from the underlying platforms that implement the model.

● Consider OAM extensions such as an OAM ‘Kind’ (an extensible component definition schema) for DICOM. The current standard definitions for Containerized Workloads may not cover the requirements of DICOM.

● Outline the expected patterns of interactions between Application and Host during the “discovery” process. Decide whether both registration (Application does an unsolicited push of Manifest to Platform) and discovery (Platform detects presence of Application and obtains Manifest from Application) are needed. Decide if endpoints that are capable of multiple functions need to provide a “list of Applications” and then Application Manifests for each.

● Choose a manifest transport mechanism (e.g. define a specific service analogous to Association Negotiation, or define a Non-Patient Instance IOD for manifests and use existing services like STOW, WADO, C-STORE, C-MOVE, etc., or extend the Part 18 Capabilities Description and use the Retrieve Capabilities Transaction, or consider other options) and describe how it might make use of the LDAP and DNS mechanisms described in Part 15.

● Decide if Platform/Host and/or Application need a new DICOM Service Class Specification, if Non Patient Instances can be used for Manifests or we may need to consider other options.

● Review the existing data exchange mechanisms between application and platform and ensure they meet the expected needs based on current platform implementation experience.

● Review all of Part 19 for coherence with the new material and update as necessary. The intention is to extend without disrupting the existing mechanisms (e.g. configuration-based registration), but if appropriate, some sections (e.g. SOAP transactions) may be deprecated.

● Review content with WG-14 (Security), WG-27 (Web), and WG-31 (Conformance)

## **Parts of Standard Affected**

● DICOM Part 2: Conformance

● DICOM Part 15: Security and System Management Profiles

● DICOM Part 18: Web Services

● DICOM Part 19: Application Hosting

And possibly

● DICOM Part 3: Information Object Definitions

● DICOM Part 4: Service Class Specifications

## **Resources & Timeline**

Brian Bialecki, ACR, will be the main editor with assistance from Chris Lindop. The Working Group will need approximately 6-12 months of bi-weekly T-cons to achieve a draft and an additional 6 months to work through public comments. Although this may be an aggressive timeline, WG-23 has been focused on these issues as it has worked on recent projects and this would be an extension of that work. Keith Houston has begun some work in this area himself that can be leveraged for this effort along with the aforementioned draft White Paper. In addition to the editors, Keith Houston (Blackford), and Brad Genereaux (NVIDIA), will be able to provide additional assistance to this effort in a limited capacity.

WG-06 resources are expected to include: 8 hours for first read with a follow up of 4 hours, 8 hours for Public Comment prep, 8 hours for Letter Ballot prep, and 4 hours for Final Text prep.