Digital Imaging and Communications in Medicine (DICOM)

Supplement 224: Service Discovery and Control

Prepared by:

DICOM Standards Committee, Working Group 23: Artificial Intelligence/Application Hosting

1300 N. 17th Street, Suite 900

Rosslyn, Virginia 22209 USA

Status: Version 4, Aug 3, 2021

Developed pursuant to DICOM Work Item 2020-08-A

Table of Contents

Table of Contents

Document History

2020/08/25	Version 1	BB	New Document – Initial Working Draft
2021/03/25	Version 2	BB	Moved Entrypoints to Traits to comply with new OAM spec
2021/06/22	Version 3	BB	Content for WG 6 first read
2021/08/03	Version 4	BB	Begin incorporating WG feedback, removing copied OAM content, instead providing references
2022/01/04	Version 5	BB	Content for WG 6 review for release to Public Comment
2022/03/15	Version 6	BB	Adjust content to remove OAM dependency

Open Issues

1.	What would be the minimum supported traits for DICOM compliance? These have been proposed in the document, Section B Table B.3.1, but are they sufficient?
2.	Should the supplement handle expired or leaked secrets, section H.2, or should they be handled by the platform? The Working Group feels this is out of scope.
3.	Fold content into Part 19. Part 19 becomes more generalized to be a collection of services and interfaces, current DICOM API becomes a nested section 9. Alternatively does this belong in Part 15, or even its own Part?
4.	For WADO should we just deal with data transfer only, and items such as iccprofile be removed? What would be the minimum attributes required to be part of the supplement? Refer to Section E.2.3
5.	Sections E.2.8 and E.2.9 were defined by the Working Group to allow operators to understand an APIs relation to the Application. Are REST API Provider and User both needed, or can Provider be sufficient to cover all use cases?
6.	Should examples, specifically Section L, be kept out of normative sections and made part 17?

Closed Issues

1.	FHIR Endpoint being used as an Ingest/Egress Method is outside the scope of this supplement, it can be created, but should not be added to DICOM.
2.	WADO-WS was left out as it is retired

1. Scope and Field of Application

Supplement 224 adds mechanisms for discovering and managing processing services which will be referred to throughout the supplement as “Applications”. Service Discovery and Control describes network services, executables, or containers as “Workloads”. The work Applications perform is described in the Application’s “Scope” which also contains the Application’s mechanism for ingestion data and delivery of results, as well as any “Options” which can be invoked. The management of Applications and the workitems they process is the responsibility of a “Platform”. Platforms register Applications and their capabilities and map those capabilities to requests. Platforms may host, instantiate or activate an Application in order to fulfill a workitem request.

2. Normative References

The following standards contain provisions that, through reference in this text, constitute provisions of this Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this Standard are encouraged to investigate the possibilities of applying the most recent editions of the standards indicated below.

Open Application Model (OAM) – v0.3.0 - https://oam.dev/

Kubernetes – v1.23 - https://kubernetes.io/

Distributed Application Runtime (Dapr) – v1.6 - https://dapr.io/

3. Definitions

Application – A combination of one or more Workloads described by an Application Definition. These can be network services, APIs, executables, or containers. The Application additionally may be hosted thus, utilizing services and resources offered by the Platform. The Application functions as the SCP to the Platform for the work to be done.

Application Definition – Used to describe an Application and it's Workload. It contains a description of the underlying implementation mechanism (service, container, etc.) as well as the Application's Scope and Option details. The Application Definition is the authoritative description of an Application's implementation and it contains the configured values at the time of initiation. Application Definitions are designed to be configured by imaging informatics staff such as a PACS administrator, allowing the Application to properly function in the hosted environment.

Application Programming Interface (API) - A set of interface methods that Applications and Platforms can use to communicate with each other.

Parameter – A Parameter is an attribute in the Scope or Workload specification that is made mutable. Parameters are required to be defined in the Application Definition.

Platform – The system used to register, manage, and interact with Applications. A Platform employs Applications to perform work, thereby functioning as the SCU for each individual Application. A Platform can host, instantiate, or use services of an Application. When the Platform acts as a host, the Platform provides the infrastructure in which the Application runs and interacts with the external environment. This includes network access including security.

Resources - Resources are individual entities, such as Secrets, Scopes, and Workloads, from which Application Definitions are composed. Each kind of Resource has a schema that describes how to describe an instance of it.

Secret – A Resource that contains a small amount of sensitive data such as a password, a token, or a key. Such information might otherwise be put in a container image. Using a Secret means that you don't need to include confidential data in your application code.

Scope – A Resource that describes a data processing operation supported by an Application. Scopes contain the Applications mechanism(s) for ingestion data and delivery of results, as well as any Options which can be invoked. Scopes are expected to be provided by clinical application vendors and reviewed by users of an Application, such as a radiologist.

Option - An Option defines a piece of add-on functionality or characteristic, that pertains to the operation of an Application and defined in the Application Scope. Options represent features of the Application that are operational in nature.

YAML - (a recursive acronym for "YAML Ain't Markup Language") is a human-readable data-serialization language. It is commonly used for configuration files and in applications where data is being stored or transmitted.

Workload – A Resource that describes an implementation mechanism such as a container or network service. Workloads mat be Containerized Workloads (which require instantiation and scaling by the Platform), Service Workloads (which provide network services or request URIs, and are not controlled or scaled by the Platform), and Executable Workloads (which are parameterized executables where scaling is the responsibility of the Platform). Workload definitions are expected to be provided by clinical application vendors and reviewed by infrastructure and/or Platform operators such as network and security engineers.

A. Service Discovery and Control Overview

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.

The concept of an Application Definition that describes the Application to a Platform is introduced. The Application Definition is a unique object that describes what does the Application does, what is needed to instantiate the Application or where it is located, as well as what the ingest and egress mechanisms for data are. The three components to an Application Definition are the Scope, the Workload and the Application's Parameter values.

A.1 Application Definition Structure Overview – YAML

The structure of a complete Application Definition YAML file is as follows:

```
Scope Definition
      Scope Specification
      Ingest Methods
      Egress Methods
      Options
      Application Definition Location
-------
Workload Definition
      Workload Specification
      Parameter Declaration
-------
Application Definition
      Scope Reference
            Parameter Values
      Workload Reference
            Parameter Values
```

The Application Definition leverages concepts from the Open Application Model (https://oam.dev/) for DICOM service discovery and control.

- A developer creates an application or service. To deliver it to users, the developer defines how to discover, instantiate, and interact with it.

    - For containerized or executable applications, this is done in a Workload(s) that encapsulate the information needed to run the application, and an associated Scope that defines how the Workload consumes and emits data.
    - For services, this is done via a complete Application that describes how to reach the application and the operations it supports.
- An Application operator deploys instances of an Application and, if necessary, configures its Parameters.
- The Application developer and Application operator have so far described an Application and its operational characteristics in platform-neutral terms. The power of this model comes from the underlying Platforms that implement the model.

This allows the three components to be used separately by their respective specialties. The Scope describes what an Application does and how it is done. This can be used to map tasks or workitems to an Application's capabilities. A Workload provides the underlying mechanism, infrastructure and properties required for instantiation. This is needed by network, security, and infrastructure engineers. The Definition is the configuration of a specific instance or instantiation of the Application. These mimic the configuration screens of the PACS Administrator in a single YAML file.

A.2 Resources

Resources are the building blocks of an Application Definition. Resources have standardized schemas defined for them. Unique instances of these Resources are defined by Application developers. Scopes and Workloads are defined prior to an Application Definition, which then references appropriate Scopes and Workloads.

When unique Scope or Workload Resources are created, those values which the owner of the Resource has made mutable, will be specified as "fromParam" in its definition. The actual values for each of these Parameters are defined as part of the Application Definition in which the Resource is used.

There are two types of Resources, base, and subordinate. Base Resources are those that can be define individually. These are Scopes, Workloads, Secret Definitions and Application Definitions. Subordinate Resources are those Resources whose scheme is referenced by type and used within a base Resource definition. Subordinate Resources are Ingest Methods, Egress Methods, and Options.

A.2.1 Base Resource Schemas

The Resource Definition schemas shall be specified as shown in Table A.2-1. All base Resources, Scopes, Workloads, Secret Definitions and Application Definitions, will use this base schema to define their contents. All attributes shall be present.

Table A.2.1-1 Resource Schema Attributes

Attribute	Type	Description
apiVersion	string	A string that identifies the version of the schema the object should have. For example, the standard types use dicomstandard.org/v1
kind	string	A string defining the type of Resource that is being defined. The types of schemas defined within this specification are applicationScope, executableWorkload, serviceWorkload, containerizedWorkload, secretDefinition and applicationDefinition.
metadata	Metadata (see A.2.1.1)	Information about the Resource. The name value in the metadata shall be unique to a Platform.
spec	Specification (see the referenced	The settings to be used to define a Resource instance. These settings are the standardized Resource schema.

	Resource schema)	

A.2.1.1 Metadata Schema

Metadata provides information about the contents of a Resource. The name attribute is used to uniquely identify the Resource when being referenced and shall be unique for the Platform. Table A.2.1.1-1 provides the schema for the metadata section of a resource. All attributes shall be present.

Table A.2.1.1-1 Resource Metadata Schema Attributes

Attribute	**Type**	**Description**
name	string	A name for the Resource instance. This name shall be unique for the Platform for the Resource type.
vendor	string	The entity that is responsible for creation of the Resource
version	string	The version for this Resource Definition

A.2.2 Subordinate Resource Schemas

The Resource definition schemas shall be specified as shown in Table A.2.2-1. All subordinate Resources, Ingest Methods, Egress Methods, and Options, will use this base schema to define their contents.

Table A.2.2-1 Resource Schema Attributes

Attribute	**Type**	**Required**	**Description**
kind	string	Y	A string defining the type of Resource that is being defined. The types of schemas defined within this specification are ingest, egress and options.
name	string	Y	A name for the Resource instance. This name shall be unique for the Platform for the Resource type.
required	array	N	A list of attributes in the Resource spec whose Parameter values must be provided for a valid Application Definition.
spec	Specification (see the referenced Resource schema)	Y	The settings to be used to define a Resource instance. These settings are the standardized Resource schema.

A.3 Example Resource Instance

An example of a Secret Definition Resource instance is shown with the recommended metadata information provided. The spec section of a Secret Definition Resource is defined in section F.1. This is how a properly formatted resource would appear.

```
# ----- Secret Definition -----
apiVersion: dicomstandard.org/v1
kind: secretDefinition
metadata:
    name: filepath-read
    vendor: mycompany
```

```
    version: 1.0
spec:
    data:
        username: read_mycompany
        passcode: MWYyZDFlMmU2N2Rm
# ----- Secret Definition -----
```

A.4 Example Subordinate Resource Instance

An example of a C-Store Ingest Method Resource instance is shown with the recommended information provided. The spec section of a C-Store Ingest Method Resource is defined in section C.1. This is how a properly formatted resource would appear.

```
ingest:
-
 kind: cstore
 name: my-c-store
 required:
 - aet
 - host
 - port
 - scu.scuAet
 - scu.scuHost
 spec:
    aet: fromParam
    host: fromParam
    port: fromParam
    tlsCertificate:
     secretKeyRef:
        name: fromParam
        key: tlscertificate
    tlsKey:
     secretKeyRef:
        name: fromParam
        key: tlskey
    scu:
    -
     scuAet: fromParam
     scuHost: fromParam
```

B. Scope Definitions

Application Scopes are used to group Workloads together into logical Applications by providing different forms of Application boundaries with common group behaviors. The schema is used to define the Scope of an Application hosted by a Platform. Scopes contain a specification section which is used by a Platform to map incoming workitems to an Applications functionality. Scopes further define an Applications data ingest and egress methods, as well as supported Application Options. Scopes will also define where the Application Definition for an instantiated instance of an Application is located.

B.1 Application Scope Specification

The following is the definition reference for Application Scope. The DICOM Application Scope is used to define an Application's capabilities. These capabilities are defined by a code system which the Platform

and Application use to process jobs. The Resource Definition itself is described in Table B.1-1 and the YAML file is listed in Annex A.

Table B.1-1 Application Scope Attributes

Attribute	Type	Required	Description
type	string	Y	Shall be workitem, route, or invoked. Details the expected responsibility of the Platform as it relates to the Application. See Section B.1.1.
codeSet	[ ]Code Set (see B.1.2)	See description	A reference to a code or set of codes defined by a terminology system. Required only for workitem type.
sopClasses	array	N	SOP Classes as defined by DICOM Part 4. This is represented as an array of SOP Classes for which the Application capabilities are valid. When an SOP Class is not present the platform may assign work to this application scope at its discretion
friendlyName	string	N	Name given to the Application to understand its basic purpose.
ingestMethods	[ ]Ingest Methods (see Section C – Ingest Methods)	N	Methods by which the Application consumes or ingests data. The kind for these Subordinate Resources shall be cstore, stow, filepath or api.
egressMethods	[ ]Egress Methods (see Section D – Egress Methods)	N	Methods by which the Application emits data or results. The kind for these Subordinate Resources shall be cstore, stow, filepath or api.
options	[ ]Options (see Section E – Application Options)	N	Defines a piece of add-on functionality or characteristic, that pertains to the operation of an Application.
auth	Secret Store (see B.1.3)	N	Location of the secret store used for security, authentication, and authorization.
definitionLocation	Definition Location (see B.1.4)	Y	Location of the Application Definition for an instantiated instance of an Application.
workloads	[ ]Workloads (see B.1.5)	Y	The Platform unique names of the Workloads used in the Application described by the Scope.

B.1.1 Operation Types

Workitem – Operations defined by a shared code system between the Application and Platform. Work items are managed by the Platform and are usually in response to job requests which may be in the form of DICOM UPS messages.

Route – Data will be sent to the Application and once successfully transferred the platform can consider the request complete.

Invoked – An Application that is invoked by some explicit user action.

B.1.2 Code Set Schema

Code Sets allow Applications to be tied to codes in their terminology system. These shall be provided when the Application Scope type is Workitem. Codes Sets are provided as YAML arrays. Table B.1.2-1 provides the schema for the Code Set section of a resource. All attributes shall be present.

Table B.1.2-1 Code Set Schema Attributes

Attribute	Type	Description
code	string	Symbol in syntax defined by the system
codeSystem	string	Identity of the terminology system

B.1.3 Secret Store Schema

Secret Stores allow Applications to securely storage and access to secrets such as access keys and passwords. Additional information on the usage and structure of Secrets can be found in Section F. When configuring attributes requiring the use of a secret store this shall be present. Table B.1.3-1 provides the schema for the Secret Store section of a resource. All attributes shall be present.

Table B.1.3-1 Secret Store Schema Attributes

Attribute	Type	Description
secretStore	string	Identifies the secret store via a representation of its primary access mechanism.

B.1.4 Definition Location Schema

The Definition Location provides the location of the Application Definition for an instantiated instance of an Application, or where a new or updated version of the Application Definition must reside upon instantiation of the Applications Workload. Table B.1.4-1 provides the schema for the Definition Location section of a resource.

Table B.1.4-1 Definition Location Schema Attributes

Attribute	Type	Required	Description
host	string	N	Host name or ip the Application uses for data transfers.
port	string	N	Port the Application uses for data transfers
path	string	Y	The endpoint, relative to the port, to which the request should be directed or URI path the file.

B.1.5 Workloads Schema

Workloads tells which Workload(s) comprise an Application. Workload Reference is provided as a YAML array, as there may be multiple Workloads that comprise a single Application. Table B.1.5-1 provides the schema for the Secret Store section of a resource. All attributes shall be present.

Table B.1.5-1 Workload Reference Schema Attributes

Attribute	Type	Description
name	string	Identifies the name of the Workload used in the Application.

B.2 Example Scope Definition Instance

An example of a Scope Definition Resource instance is shown. The Specification section includes Ingest Methods from Section C, Egress Methods from Section D and Options from Section F. This is how a properly formatted resource would appear.

```
# ----- Scope Definition -----
apiVersion: dicomstandard.org/v1
kind: applicationScope
metadata:
    name: my-job-scope
    vendor: mycompany
    version: 1.0
spec:
    type: workitem
    codeSet:
    -
     code: RDES128
     codeSystem: https://radelement.org
    -
     code: 449796001
     codeSystem: https://www.snomed.org/
    sopClasses:
    - 1.2.840.10008.5.1.4.1.1.2
    - 1.2.840.10008.5.1.4.1.1.2.1
    - 1.2.840.10008.5.1.4.1.1.4
    - 1.2.840.10008.5.1.4.1.1.4.1
    friendlyName: SubduralHematomaDetection
    ingest:
    -
     kind: cstore
     name: my-c-store
     required:
     - aet
     - host
     - port
     - scu.scuAet
     - scu.scuHost
     spec:
        aet: fromParam
        host: fromParam
        port: fromParam
        tlsCertificate:
         secretKeyRef:
            name: fromParam
            key: tlscertificate
        tlsKey:
         secretKeyRef:
            name: fromParam
            key: tlskey
        scu:
        -
         scuAet: fromParam
         scuHost: fromParam
    -
     kind: filepath
     name: my-filePath
```

```
 spec:
    path: fromParam
    username:
     secretKeyRef:
        name: fromParam
        key: username
    passcode:
     secretKeyRef:
        name: fromParam
        key: passcode
egress:
-
 kind: cstore
 name: your-c-store
 required:
 - destAet
 - destHost
 - destPort
 spec:
    destAet: fromParam
    destHost: fromParam
    destPort: fromParam
    tlsCertificate:
     secretKeyRef:
        name: fromParam
        key: tlscertificate
    tlsKey:
     secretKeyRef:
        name: fromParam
        key: tlskey
-
 kind: filepath
 name: your-filePath
 required:
 - path
 spec:
    path: fromParam
    username:
     secretKeyRef:
        name: fromParam
        key: username
    passcode:
     secretKeyRef:
        name: fromParam
        key: passcode
options:
-
 kind: license
 name: my-app-lic
 required:
 - licenseKey
 - machineCode
 spec:
    licenseKey: fromParam
    machineCode: yourmachineid123
-
 kind: auditTrail
```

```
     name: my-app-audit
     required:
     -syslogUri
     spec:
        syslogUri: fromParam
        tlsCertificate:
         secretKeyRef:
            name: fromParam
            key: tlscertificate
        tlsKey:
         secretKeyRef:
            name: fromParam
            key: tlskey
    auth:
        secretStore: fromParam
    definitionLocation:
        host: fromParam
        port: 8080
        path: /appdef
    workloads:
        -
        name: my-appserver
# ----- Scope Definition -----
```

C. Ingest Methods

Ingest Methods state how the Application can interact with data and systems for ingestion. Ingest Methods are defined in the Application Scope and are provided as an array. There is no defined mapping between Ingest and Egress Methods as this an implementation detail outside the scope of this standard.

The Ingest Methods described by DICOM are:

- cstore – See section C.1
- stow – See section C.2
- filepath – See section C.3
- api – See section C.4

These shall be listed in the attribute kind for this Subordinate Resource.

C.1 C-Store

This method is used to specify DIMSE endpoint information for an Application acting as a C-Store SCP. Table C.1-1 provides the attributes for the C-Store Ingest Method.

Table C.1-1 C-Store Ingest Method Attributes

Attribute	Type	Required	Description
aet	string	Y	AET the Application uses for inbound data transfers, the Application's storage SCP AET.
host	string	Y	Host name or ip the Application uses for inbound data transfers.
port	integer	Y	Port the Application uses for inbound data transfers.

tlsCertificate	string	N	Trusted certificate for this communication. This should be communicated with a Secret when used.
tlsKey	string	N	Key used to decrypt data. This should be communicated with a Secret when used.
scuAet	string	N	AET of SCU if required by SCP.
scuHost	string	N	Host name or ip of SCU if required by SCP.

C.1.1 Example C-Store Ingest Method

```
ingest:
-
 kind: cstore
 name: my-c-store
 required:
 - aet
 - host
 - port
 - scu.scuAet
 - scu.scuHost
 spec:
   aet: fromParam
   host: fromParam
   port: fromParam
   tlsCertificate:
    secretKeyRef:
      name: fromParam
      key: tlscertificate
   tlsKey:
    secretKeyRef:
      name: fromParam
      key: tlskey
   scu:
   -
    scuAet: fromParam
    scuHost: fromParam
```

C.2 STOW

This method is used to specify DICOMweb endpoint information for an Application acting as a Provider. These attributes may apply to Ingest or Egress Methods. Table C.2-1 provides the attributes for STOW Methods.

Table C.2-1 STOW Provider Method Attributes

Attribute	**Type**	**Required**	**Description**
resourceUri	string	Y	Identifies a resource via a representation of its primary access mechanism.
contentTypeHeaders	string	Y	Section 8.7.3.5 DICOM Media Type Syntax

tlsCertificate	string	N	Trusted certificate for this communication. This should be communicated with a Secret when used.
tlsKey	string	N	Key used to decrypt data. This should be communicated with a Secret when used.

C.2.1 Example STOW Provider Method

```
-
  kind: stow
  name: my-stow
  required:
  - resourceUri
  spec:
    resourceUri: fromParam
    contentTypeHeaders: "application/dicom"
    tlsCertificate:
     secretKeyRef:
       name: fromParam
       key: tlscertificate
    tlsKey:
     secretKeyRef:
       name: fromParam
       key: tlskey
```

C.3 File Path

When using the File Path Ingest Method, data shall be populated at the time of job start for task as opposed to a monitored folder. Data Types (SOP classes) are defined as part of the Application's Scope when DICOM files are used as the input type. When used for data types other than DICOM, data types can be listed here, although the content and coordination of formats are outside the scope of this specification and need to be coordinated between the Platform and Application. Applications will have a minimum of read permission granted to a file path for data ingest. The file path is the path to the data which is to be used to perform the requestion action. This differs from paths specified in a Containerized Workload as these as specifically used to pass data for job execution.

File path is used to specify information for an Application acting as a user is data contained at the path. Table C.3-1 provides the attributes for the File Path Method.

Table C.3-1 File Path Method Attributes

Attribute	**Type**	**Required**	**Description**
path	string	Y	To the folder level. Everything in the folder will be ingested when specified as an Ingest Method. The subfolder structure is not specified for Egress Methods.
userType	string	N	Shall be basic or userIdPasscode
username	string	N	Identification used to access resource if required. This should be communicated with a Secret when used.
passcode	string	N	Passcode used to access resources if required. This should be communicated with a Secret when used.

dataTypes	array	N	When not present MIME types are considered to be DICOM and are configured as part of the scope via SOP classes.
efsAlgorithm	string	N	The symmetric encryption algorithm used. This should be communicated with a Secret when used.
efsKey	string	N	Used to decrypt data if required. This should be communicated with a Secret when used.

C.3.1 Example File Path Method

```
-
 kind: filepath
 name: your-filePath
 required:
 - path
 spec:
    path: fromParam
    username:
     secretKeyRef:
        name: fromParam
        key: username
    passcode:
     secretKeyRef:
        name: fromParam
        key: passcode
```

C.4 API

The API Method is used to specify API endpoint information for an Application acting as an API Provider. Note the specific API specification is not configured here, as this is just a reference to the API itself. The API may then expose additional entrypoints or services beyond the scope of this specification as part of its own specification. Table C.4-1 provides the attributes for the API Provider Method.

Table C.4-1 API Provider Method Attributes

Attribute	**Type**	**Required**	**Description**
apiName	string	Y	Examples are AcrModelApi, DicomWSDL
apiVersion	string	Y	A string that identifies the version of the API
resourceUri	string	Y	Identifies a resource via a representation of its primary access mechanism.
uriType	string	Y	Examples are request, liveliness, readiness, log
tlsCertificate	string	N	Trusted certificate for this communication. This should be communicated with a Secret when used.
tlsKey	string	N	Key used to decrypt data. This should be communicated with a Secret when used.
authMethod	string	N	basicAuth, formAuth, clientCertAuth, oAuth, bearerAuth
apiKey	string	N	Key used to connect to the API. This should be communicated with a Secret when used.
accessToken	string	N	The authorization of a specific application. This should be communicated with a Secret when used.

refreshToken	string	N	Used to acquire new access token. This should be communicated with a Secret when used.

C.4.1 Example API Provider Method

```
-
 kind: api
 name: your-dicom-api
 required:
 - resourceUri
 spec:
     apiName: DicomWSDL
     apiVersion: 2021b
     resourceUri: fromParam
     uriType: request
```

D. Egress Methods

Egress Methods state how the Application can interact with data and systems for communication of results or artifacts. Egress Methods are defined in the Application Scope and are provided as an array. There is no defined mapping between Ingest and Egress Methods as this an implementation detail outside the scope of this standard.

The Egress Methods described by DICOM are:

- cstore – See section D.1
- stow – See section C.2
- filepath – See section D.2
- api – See section C.4

These shall be listed in the attribute kind for this Subordinate Resource.

D.1 C-Store

This method is used to specify DIMSE endpoint information for an application acting as a C-Store SCU. Table D.1-1 provides the attributes for the C-Store Egress method.

Table D.1-1 C-Store Egress Method Attributes

Attribute	Type	Required	Description
destAet	string	Y	AET the Application uses for outbound data transfers, the external system's storage SCP AET.
destHost	string	Y	Host name or ip the Application uses for outbound data transfers.
destPort	integer	Y	Port the Application uses for outbound data transfers.
tlsCertificate	string	N	Trusted certificate for this communication. This should be communicated with a Secret when used.
tlsKey	string	N	Key used to decrypt data. This should be communicated with a Secret when used.

D.1.1 Example C-Store Egress Method

```
egress:
-
 kind: cstore
 name: your-c-store
 required:
 - destAet
 - destHost
 - destPort
 spec:
   destAet: fromParam
   destHost: fromParam
   destPort: fromParam
   tlsCertificate:
    secretKeyRef:
      name: fromParam
      key: tlscertificate
   tlsKey:
    secretKeyRef:
      name: fromParam
      key: tlskey
```

D.2 File Path

Applications and their Workloads shall have a minimum of read/write permission granted to a file path for data egress. This allows for writing artifacts as well as verifying their existence and that they are not corrupt. The egress file path is the path to where the data artifacts are to be placed after the Application performs the requestion action. This differs from paths specified in a Containerized Workload as these as specifically used to pass data artifacts resulting from job execution.

File path is used to specify information for an Application acting as a user is data contained at the path. Table C.3-1 provides the attributes for the File Path Method.

E. Application Options

An Option defines a piece of add-on functionality or characteristic, that pertains to the operation of an Application and defined in the Application Scope. Options represent features of the Application that are operational in nature.

The Options described by DICOM are:

- license
- auditTrail
- jobTimeout
- timeSync
- scaler

These shall be listed in the attribute kind for this Subordinate Resource.

E.1 License

This Option is used to specify License information for an Application. Table E.1-1 provides the attributes for the License Option.

Table E.1-1 License Option Attributes

Attribute	Type	Required	Description
licenseKey	string	Y	Application defined license key string
machineKey	string	N	Machine specific code generated, example MAC or some other machine code

E.1.1 Example License Option Usage

```
options:
-
 kind: license
 name: my-app-lic
 required:
 - licenseKey
 - machineCode
 spec:
    licenseKey: fromParam
    machineCode: yourmachineid123
```

E.2 Audit Trail

The Option specified in this section provides information on Audit Trail endpoints for connectivity only, not for content. For information on Audit Trial Message formats, schemas and coding refer to PS3.15 Annex A. Table E.2-1 provides the attributes for the Audit Trail Option.

Table E.2-1 Audit Trail Option Attributes

Attribute	Type	Required	Description
syslogUri	string	Y	Identifies the resource by name at the specified location or URL
tlsCertificate	string	N	Trusted certificate for this communication. This should be communicated with a Secret when used.
tlsKey	string	N	Key used to decrypt data. This should be communicated with a Secret when used.

E.2.1 Example Audit Trail Option Usage

```
options:
-
 kind: auditTrail
 name: my-app-audit
 required:
 -syslogUri
 spec:
    syslogUri: fromParam
    tlsCertificate:
```

```
    secretKeyRef:
       name: fromParam
       key: tlscertificate
   tlsKey:
    secretKeyRef:
       name: fromParam
       key: tlskey
```

E.3 Job Timeout

This Option is used to set a timeout for a job. Once the timeout is exceeded the status will become failed with reason timeout exceeded. If the Platform is controlling the container or task the Platform may terminate the instance. Table E.3-1 provides the attributes for the Job Timeout Option. All attributes shall be present.

Table E.3-1 Job Timeout Option Attributes

Attribute	Type	Description
seconds	integer	Used to set a timeout for a job. If included and not made mutable, the default value of this attribute shall be 1.

E.3.1 Example Job Timeout Option Usage

```
options:
-
 kind: jobTimeout
 name: my-job-timeout
 required:
 -seconds
 spec:
    seconds: 60
```

E.4 Time Sync

This Option is used to ensure events are in synchronization, the use of a common time synchronization server is commonly used. Table E.4-1 provides the attributes for the Time Sync Option. All attributes shall be present.

Table E.4-1 Time Sync Option Attributes

Attribute	Type	Description
ntpTimeServer	string	Server address used for time synchronization.

E.4.1 Example Time Sync Option Usage

```
options:
-
 kind: timeSync
 name: my-time-sync
 required:
 - ntpTimeServer
 spec:
```

```
    ntpTimeServer: time.nist.gov
```

E.5 Scaler

This Option allows operators to manually scale the number of replicas for Workload types that allow scaling operations. Table E.5-1 provides the attributes for the Scaler Option. All attributes shall be present.

Table E.5-1 Scaler Option Attributes

Attribute	Type	Description
replicaCount	integer	The target number of replicas to create for a given Application instance. If included and not made mutable, the default value of this attribute shall be 0, whereby no replica would be created.

E.5.1 Example Scaler Option Usage

```
options:
-
 kind: scaler
 name: my-app-scaler
 required:
 - replicaCount
 spec:
    replicaCount: 1
```

In this example two (2) instances of the Application's Workload would be instantiated.

F. Secrets and Security

This section covers the implementation and operational security aspects within a deployment. While the use of HTTPS is the only requirement, this section provides guidance on additional mechanisms for securing Applications. Additionally, implementers shall consider BCP 195 TLS Secure Transport Connection Profile, Basic Application Level Confidentiality Profile, and other DICOM profiles which are relevant to the Application.

F.1 Securing API Communication

All API communications for registration and discovery shall enforce the use of HTTPS. Resources used in DICOM Applications contain sensitive information such as passwords, tokens, encryption keys and certificates. Additionally, running Applications may exchange or access DICOM objects which contain demographic and medical information about the patient which need to be kept confidential. Both require a secure communications channel.

Developers of DICOM Applications should consider how secrets are exchanged within their implementations.

Note: The Distributed Application Runtime (Dapr), (https://dapr.io/) is an example of such an implementation that provides best practices on security such as the use of TLS, OAuth and token authentication.

F.2 Secrets

```
# ----- Secret Definition -----
apiVersion: dicomstandard.org/v1
kind: secretDefinition
metadata:
    name: cstore-tls
    vendor: mycompany
    version: 1.0
data:
    # the data is abbreviated in this example
    tlscertificate: MIIC2DCCAcCgAwIBAgIBATANBgkqh ...
    tlskey: MIIEpgIBAAKCAQEA7yn3bRHQ5FHMQ ...
# ----- Secret Definition -----
```

G. Workloads

A Workload is an entity that maps characteristics to an implementation mechanism such as a container or network service that are needed by network, security, and infrastructure engineers. These can be Containerized Workloads, which requires instantiation and scaling is the responsibility of the Platform, Service Workloads, which provides network services or request URIs, are not controlled by the Platform and scaling is the responsibility of the Workload, and Executable Workloads, which are parameterized executables where scaling is the responsibility of the Platform.

G.1 Executable Workload Specification

An Executable Workload provides a command path and its required environmental parameters. Executable Workloads are used to describe parameterized executables which can be local or remote, if environment is hard to reproduce, it can be created on a remote server and called from there. Table G.1-1 provides the schematic specification for an Executable Workload.

Table G.1-1 Executable Workload Attributes

Attribute	Type	Required	Description
exec	Command (see Section G.1.1)	Y	The path or uri to the executable.
env	[ ]Env (see Section G.1.2)	N	Environment variables. For Executable Workload types environmental variables such as operating system or runtime component requirements should be specified here.
parameters	[ ]Parameters (see Section G.1.2)	N	Allows the owner of the Resource to make values mutable or required based on the environment where the Workload is executed.

G.1.1 Command

Table G.1.1-1 provides the attributes for the command section of an Executable Workload. All attributes shall be present.

Table G.1.1-1 Command Schema Attributes

Attribute	Type	Description
command	[ ]string	A command to be executed. Each command will be executed sequentially. Commands exiting 0 are considered successful.

G.1.2 Env

Env describes an environment variable as a name/value pair of strings. Table G.1.2-1 provides the attributes for the env section of an Executable Workload. All attributes shall be present.

Table G.1.2-1 Env Schema Attributes

Attribute	Type	Description
name	string	The environment variable name. Must be unique per Application.
value	string	The environment variable value.

G.1.3 Parameters

Env describes an environment variable as a name/value pair of strings. Table G.1.2-1 provides the attributes for the env section of an Executable Workload. All attributes shall be present.

Table G.1.3-1 Parameters Schema Attributes

Attribute	Type	Description
name	string	The parameter name to be referenced in the Application Definition. Must be unique per Application.
required	boolean	Determines whether a value needs to be provided in the Application Definition for the Application Definition to be valid.

G.1.4 Example Executable Workload Specification

```
# ----- Workload Definition -----
apiVersion: dicomstandard.org/v1
kind: containerizedWorkload
metadata:
    name: my-executable
    vendor: mycompany
    version: 1.0
spec:
    exec:
       command:
       - \\myshare.mynetwork.org\programs\exampletask\task.jar
    env:
       - name: jreVersion
         value: 1.8
# ----- Workload Definition -----
```

G.2 Service Workload Specification

A Service Workload is used to describe persistent network services or APIs that need not be instantiated by the Platform. Table G.2-1 provides the schematic specification for Service Workload.

Table G.2-1 Service Workload Attributes

Attribute	Type	Required	Description
host	string	Y	Host name or ip of the server.
livenessProbe	HealthProbe (see Section G.2.1)	N	Instructions for assessing whether the server is alive.
readinessProbe	HealthProbe (see Section G.2.1)	N	Instructions for assessing whether the server is in a suitable state to serve traffic.
Parameters	[ ]Parameters (see Section G.1.2)	N	Allows the owner of the Resource to make values mutable or required based on the environment where the Workload is hosted.

G.2.1 Health Probe

Health Probe describes how a probing operation is to be executed as a way of determining the health of a service. Table G.2.1-1 provides the schematic specification for Health Probe.

Table G.2.1-1 Health Probe Attributes

Attribute	Type	Required	Description
exec	Command (see Section G.1.1)	N	Instructions for assessing container health by executing a command. Either this attribute or the httpGet attribute or the tcpSocket attribute shall be specified. This attribute is mutually exclusive with both the httpGet attribute and the tcpSocket attribute.
httpGet	HTTPGet (see Section G.2.1.1)	N	Instructions for assessing container health by executing an HTTP GET request. Either this attribute or the exec attribute or the tcpSocket attribute shall be specified. This attribute is mutually exclusive with both the exec attribute and the tcpSocket attribute.
tcpSocket	TCPSocket (see Section G.2.1.2)	N	Instructions for assessing container health by probing a TCP socket. Either this attribute or the exec attribute or the httpGet attribute shall be specified. This attribute is mutually exclusive with both the exec attribute and the httpGet attribute.

initialDelaySeconds	integer	N	Number of seconds after the container is started before the first probe is initiated. If included and not made mutable, the default value of this attribute shall be 0.
periodSeconds	integer	N	How often, in seconds, to execute the probe. If included and not made mutable, the default value of this attribute shall be 10.
timeoutSeconds	integer	N	Number of seconds after which the probe times out. If included and not made mutable, the default value of this attribute shall be 1.
successThreshold	integer	N	Minimum consecutive successes for the probe to be considered successful after having failed. If included and not made mutable, the default value of this attribute shall be 1.
failureThreshold	integer	N	Number of consecutive failures required to determine the container is not alive (liveness probe) or not ready (readiness probe). If included and not made mutable, the default value of this attribute shall be 3.

G.2.1.1 HTTP Get

Table G.2.1.1-1 provides the schematic specification for HTTP Get.

Table G.2.1.1-1 HTTP Get Attributes

Attribute	**Type**	**Required**	**Description**
path	string	Y	The endpoint, relative to the port, to which the HTTP GET request should be directed.
Port	integer	Y	The TCP socket within to which the HTTP GET request should be directed.
httpHeaders	[ ]HTTPHeader (see Section G.2.1.1.1)	N	Optional HTTP headers.

G.2.1.1.1 HTTP Header

Table G.2.1.1.1-1 provides the schematic specification for HTTP Header. All attributes shall be present. Both name and value must abide by the HTTP/1.1 specification for valid header values.

Table G.2.1.1.1-1 HTTP Header Attributes

Attribute	**Type**	**Description**
name	string	An HTTP header name. This must be unique per HTTP GET-based probe.
Value	string	An HTTP header value.

G.2.1.2 TCP Socket

Table G.2.1.2-1 provides the schematic specification for TCP Socket. All attributes shall be present. Port must be an integer value greater than 0.

Table G.2.1.2-1 TCP Socket Attributes

Attribute	Type	Description
port	integer	The TCP socket within the container that should be probed to assess container health.

G.2.2 Example Service Workload Specification

```
# ----- Workload Definition -----
apiVersion: dicomstandard.org/v1
kind: serviceWorkload
metadata:
  name: example-dicom-server
spec:
  host: mydicomserver.myhospital.org
  livenessProbe:
    httpGet:
      port: 8086
      path: /healthz
  readinessProbe:
    httpGet:
      port: 8088
      path: /healthz
# ----- Workload Definition -----
```

G.3 Containerized Workload Specification

A Containerized Workload is a Serverless Container style workload definition that could be referenced as the schema for long-running containerized workload types for runtime platforms like Azure ACI, AWS Fargate or simple stateless workload of Kubernetes. Containerized Workloads provide the orchestration templates for a compatible Platform to properly instantiate them. Table G.3-1 provides the schematic specification for a Containerized Workload.

Table G.3-1 Containerized Workload Attributes

Attribute	Type	Required	Description
type	string	Y	Informs the Platform what schema should be expected in the Workload's schema attribute and the mechanism by which the container is instantiated. For example, docker, oam or kubevela.
Version	string	Y	This version refers to the version for the type of Containerized Workload. For example, in OAM this may be v.0.3.0
schema	Schema (see Section G.3.1)	Y	Defined by the type and version of the Workload Specification.
Parameters	[ ]Parameters (array) (see Section G.1.2)	N	Allows the owner of the Resource to make values mutable or required based on the environment where the Workload is executed.

G.3.1 Schema

DICOM makes no claim to support or provide preference for any container schema. This set of attributes provides a framework for any container specification schema that the host Platform may support for an Application. Containers schemas are defined by their type i.e., OAM, Docker, Helm, Kube.

G.3.2 Example Containerized Workload Specification

```
# ----- Workload Definition -----
apiVersion: dicomstandard.org/v1
kind: containerizedWorkload
metadata:
    name: my-appserver
    vendor: mycompany
    version: 1.0
spec:
    type: oam
    version: v1alpha2
    scheme:
        apiVersion: core.oam.dev/v1alpha2
        kind: ApplicationConfiguration
        metadata:
            name: webserver-example
        spec:
          components:
          - componentName: example-server
            parameterValues:
            - name: instanceName
              value: cool-example
            - name: cacheSecret
              value: cache-connection
            traits:
            - trait:
                apiVersion: core.oam.dev/v1alpha2
                kind: ManualScalerTrait
                spec:
                  replicaCount: 3
            scopes:
            - scopeRef:
                apiVersion: core.oam.dev/v1alpha2
                kind: NetworkScope
                name: example-vpc-network
          - componentName: example-cache
            parameterValues:
            - name: instanceName
              value: cool-example
            - name: engineVersion
              value: "4.0"
            - name: secret
              value: cache-connection
    parameters:
    - name: dhcpServer
      Required: true
# ----- Workload Definition -----
```

H. Application Definition

The Application Definition is the authoritative description of an Application's implementation and it contains the configured values at the time of initiation. Application Definitions are designed to be configured by imaging informatics staff such as a PACS administrator, allowing the Application to properly function in the hosted environment. The Application Definition can be seen as a computer readable, standardized configuration for DICOM Applications.

H.1 Application Definition Specification

Table H.1-1 Application Definition Attributes

Attribute	Type	Description
scopeRef	Reference (see Section H.1.1)	
workloadRef	[ ] Reference (see Section H.1.1)	

H.1.1 Reference

Table H.1.1-1 Reference Attributes

Attribute	Type	Description
name	string	
parameterValues	[ ] Parameter Values (see Section H.1.1.1)	

H.1.1.1 Parameter Values

Table H.1.1.1-1 Parameter Values Attributes

Attribute	Type	Description
path	string	
value	string	

H.2 Example Application Definition Instance

```
# ----- Application Definition -----
apiVersion: dicomstandard.org/v1
kind: applicationDefinition
```

```
metadata:
    name: my-app
    vendor: mycompany
    version: 1.0
spec:
    scopeRef:
       name: my-job-scope
       parameterValues:
       - path: ingest[my-c-store].spec.aet
         value: MYAPPAET
       - path: ingest[my-c-store].spec.host
         value: myapp.myhospital.org
       - path: ingest[my-c-store].spec.port
         value: 104
       - path: ingest[my-c-store].spec.tlsCertificate.secretKeyRef.name
         value: cstore-tls
       - path: ingest[my-c-store].spec.tlsKey.secretKeyRef.name
         value: cstore-tls
       - path: ingest[my-c-store].spec.scu.scuAet
         value: MYPACSAET
       - path: ingest[my-c-store].spec.scu.scuHost
         value: mypacs.myhospital.org
       - path: ingest[my-filePath].spec.path
         value: \\fileshares.myhospital.org\myapp\data\in
       - path: ingest[my-filePath].spec.username.secretKeyRef.name
         value: filepath-read
       - path: ingest[my-filePath].spec.passcode.secretKeyRef.name
         value: filepath-read
       - path: egress[your-c-store].spec.aet
         value: MYVNAAET
       - path: egress[your-c-store].spec.host
         value: myvna.myhospital.org
       - path: egress[your-c-store].spec.port
         value: 104
       - path: egress[your-c-store].spec.tlsCertificate.secretKeyRef.name
         value: vna-tls
       - path: egress[your-c-store].spec.tlsKey.secretKeyRef.name
         value: vna-tls
       - path: egress[your-filePath].spec.path
         value: \\fileshares.myhospital.org\myapp\data\in
       - path: egress[your-filePath].spec.username.secretKeyRef.name
         value: filepath-write
       - path: egress[your-filePath].spec.passcode.secretKeyRef.name
         value: filepath-write
       - path: options[my-app-lic].spec.licenseKey
         value: dui2p3jdoj28jd
       - path: options[my-app-audit].spec.syslogUri
         value: \\logs.myhospital.org\myapp\
       - path: options[my-app-audit].spec.tlsCertificate.secretKeyRef.name
         value: syslog-tls
       - path: options[my-app-audit].spec.tlsKey.secretKeyRef.name
         value: syslog-tls
       - path: auth.secretStore
         value: https://mysecrets.myhospital.org:8443
       - path: definitionLocation.host
         value: myapp.myhospital.org
    -
```

```
    workloadRef:
       name: my-appserver
       parameterValues:
       - name: dhcpServer
         value: dhcp.myhospital.org
# ----- Application Definition -----
```

I. Registration

Registration involves the process by which Platforms become aware of Applications. These Applications can exist as services or APIs, or be invoked through instantiation, executable or a combination of methods. Applications need a Platform that supports its Workloads as well as its Ingest and Egress requirements. Platforms should publish the Workloads, Option as well as Ingest and Egress Methods they support or can proxy as part of their DICOM Conformance Statement. Platforms should also publish the code system(s) that are supported to describe Application capabilities as these need to be mapped or presented as part of the Application Scope if they wish to automatically register Applications.

J. Discovery

K. Control

L. Workflow