Table of Contents

Glossary

0SAS-ESC Peering Protocol

This document specifies the Application Programming Interface (API) for Interface between Spectrum Access System (SAS) and Environmental Sensing Capability (ESC).

Prerequisite operations

The following operations shall be done before SAS-ESC communications.

1. Exchange of Public Key Infrastructure (PKI) between SAS Administrator and ESC Operator
2. Exchange of Private Key for digital signature
3. Setting of communication timeout

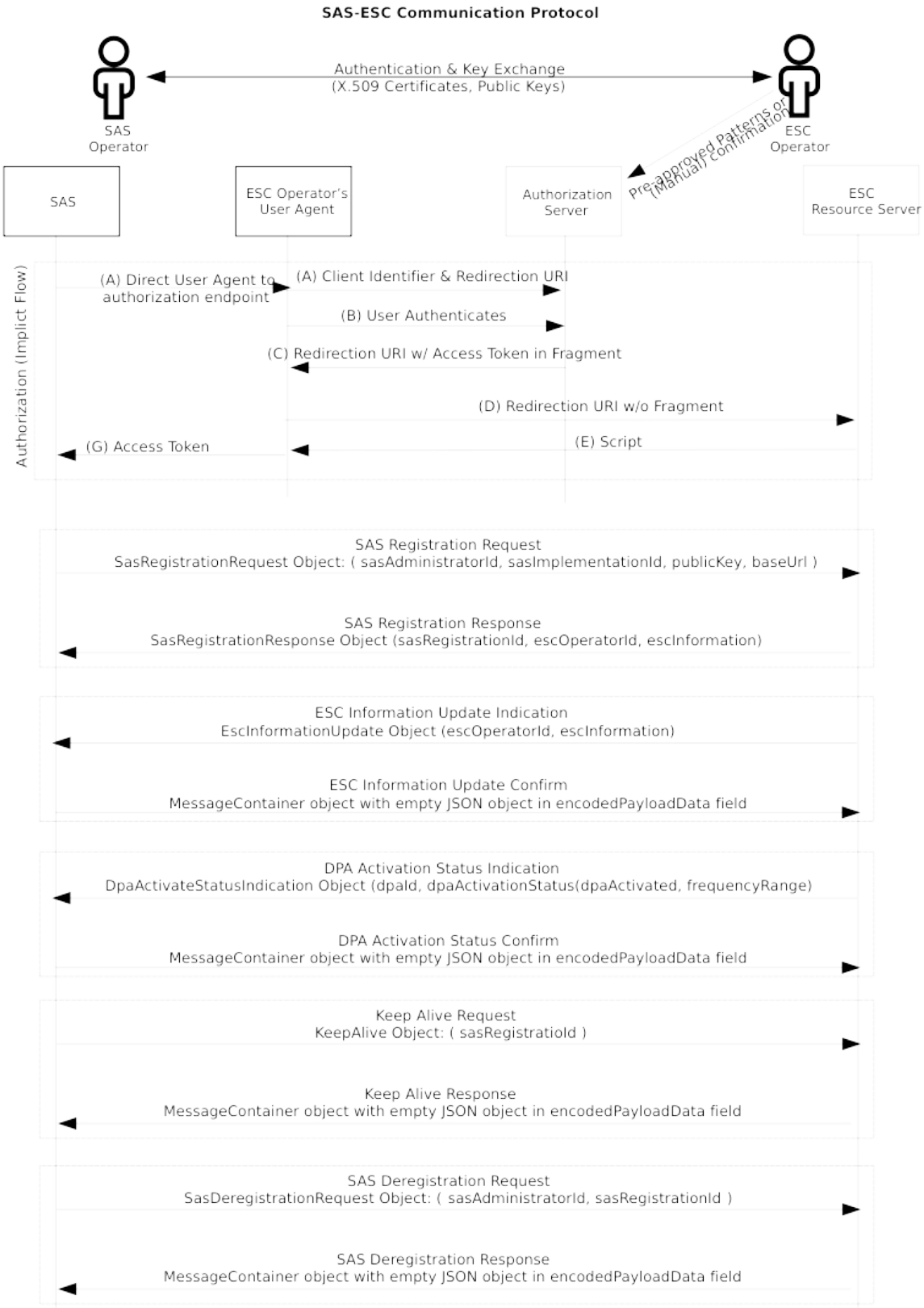


Figure: SAS-ESC Communication Life Cycle

0.1Independence of SAS and ESC

Key Bridge expects the FCC to authorize more than one party to operate a ESC, and that the internal architecture and configuration of each ESC may differ. Key Bridge further expects that each ESC, or at least the ESC in this proposal, will be independent and autonomous to the one or more SASs that depend upon their respective ESC for incumbent detection. The same principle applies in reverse as well: a SAS's internal architecture and configuration is also independent and autonomous to its ESC, and each ESC may pursue different strategies and implement different technologies for incumbent detection.

Regardless of their respective internal architecture, technology and detection strategies, a ESC and SAS must interface to share information about spectrum availability or un-availability. This essential concept is called peering and is shown in the illustration below. Through peering a SAS and ESC may exchange data in a neutral manner irrespective of their internal architecture and configuration.

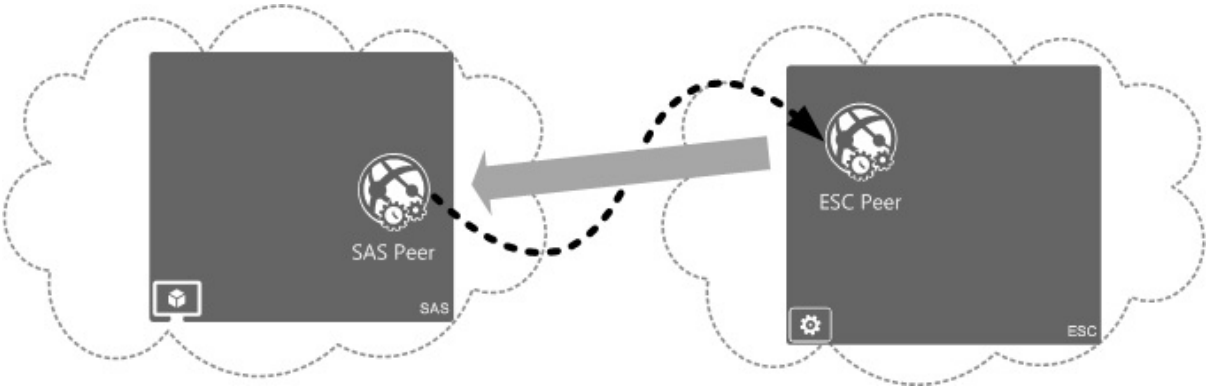


Illustration 1: SAS to ESC Peering masks internal architecture.

1.1SAS-ESC Peering Protocol

A SAS exterior peering protocol standard will place no constraint on the internal architecture and operation of the various entities that use the protocol. Referring again to Illustration 1, a generic peering process enables SAS of unknown constitution on the left peers with a Key Bridge ESC on the right.

> **This paragraph needs to be reviewed** SAS to ESC peering operational security policies and configurations are under development within a multi-stakeholder group in which Key Bridge is a participant (The Wireless Innovation Forum, Spectrum Sharing Committee, Working Group 2: Security). The actual protocol and data structures for SAS to ESC peering is not the subject of multi-stakeholder group consideration. Key Bridge has therefore invented proprietary and also proposed non-proprietary methods that a SAS may use when coordinating with a ESC to dynamically protect a non-informing incumbent spectrum user. The Key Bridge SAS Gateway Protocol is, at present, a proprietary application peering protocol that includes mechanisms to support cryptographically secure message exchange for both SAS to SAS and SAS to ESC peering, including all security prescriptions identified by the multi-stakeholder group.

A SAS and ESC Peering relationship using the SAS Gateway Protocol is secure and neutral. A SAS Infrastructure may register with and receive information from any inter-operable ESC Infrastructure supporting a peering service. When using the Key Bridge SAS Gateway Protocol a compatible SAS may register with and receive information from a compatible ESC via a designated ESC Peering API.

In the Key Bridge architecture ESC Service Nodes are responsible for Non-Informing Incumbent User (NIIU) detection services and peer directly with a subscribing SAS via the SAS Gateway Protocol. This strategy is designed to limit the geographic extent of NIIU information conveyed to a SAS to that necessary for the SAS to administer CBSDs in a specific geographic region. However, the ESC places no constraint on SAS internal architecture or operation, and a SAS may peer with multiple ESC Service Nodes.

2.1Generic SAS Peering Workflow

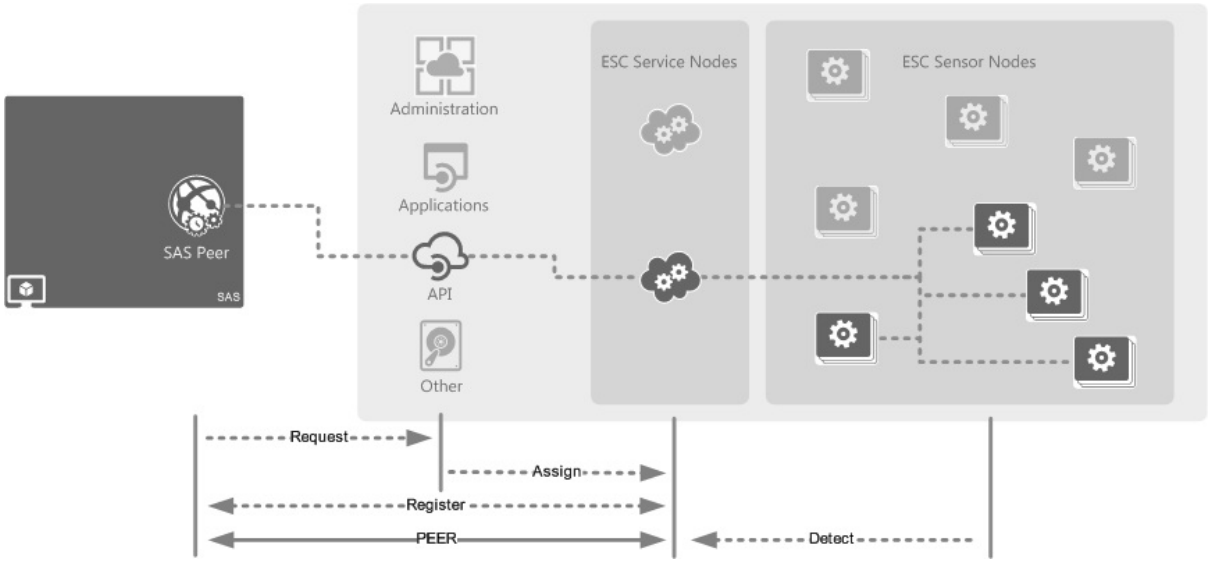


Illustration 2: ESC Service Nodes provide Non-Informing Incumbent User (NIIU) detection services to a SAS.

The process of a generic SAS peering with the Key Bridge ESC Infrastructure is detailed in Illustration 2, and proceeds according to the following general strategy.

- **Request**. A SAS first inquires about service availability to the ESC Infrastructure via a general peering API interface. The SAS service request includes the geographic area or areas where the SAS wishes to provide spectrum administration. If ESC services are not available in the requested geographic area the request is rejected (i.e. fails), otherwise the request is assigned to a matching ESC Service Node.
- **Assignment**. If NIIU detection capability is available in the SAS requested geographic area the ESC Infrastructure will assign the SAS registration request to a matching ESC Service Node.
- **Register**. After a geographic match has been established the respective ESC Service Node and the SAS exchange detailed information necessary to establish a direct and persistent peering relationship. The registration process includes conveying to the ESC Peer the specific SAS geographic regions of responsibility plus instructions describing how the ESC may query the SAS for additional information and also how to notify the SAS of ESC spectrum availability information, updates and instructions.
- **Peer**. Once established the ESC Service Node will respond to spectrum availability requests within its geographic area of service and also inform the SAS about any changes in spectrum availability.

An artifact of this CONOPS is that the Key Bridge ESC Service Nodes are assigned responsibility to determine the quantity and quality of protection for the incumbent user within their ESC Service Area, while responsibility to effect that protection is retained within a SAS. That is to say: An ESC Service Node is the service providing component of the ESC Infrastructure, and it is the ESC Service Node that authorizes or de-authorizes CBSD operations within their responsible service area according to a *geographic partitioning strategy* within their respective ESC Service Area.

4SAS Client Authentication

Proir to SAS-ESC communications, following operations need to occur:

1. Exchange of Public Key Infrastructure (PKI) between SAS Administrator and ESC Operator
2. Exchange of Private Key for digital signature
3. Setting of communication timeout

Item 1. is outlined in Figure 1 and explained below.

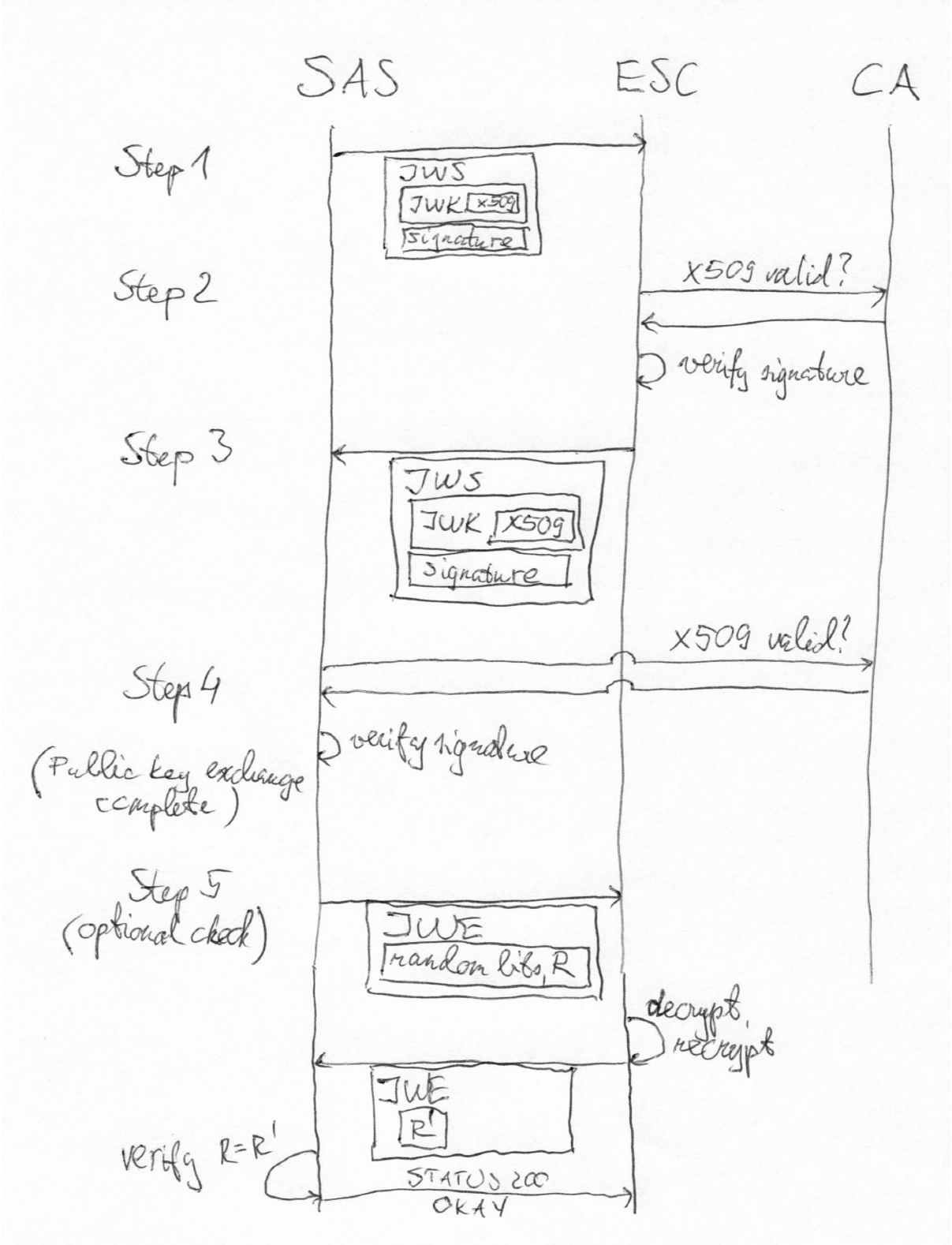


Diagram 1. SAS-ESC public key exchange

Step 0 One ESC, *E*, and one SAS, *S*, are in operation. E knows nothing of S, whereas S knows E's URL which has been provided to S by S's operators and the ESC API. Both partes (E and S) have their respective X.509 certificates in possession.

Step 1 S sends its public key to E as a JSON Web Signature which in the payload contains the pulic key in the JSON Web Key format (RFC 7517). The JWK must contain sufficient information for E to validate S's obtain S's X.509 certificate (key_e must contain a valid `x5u` field with a URL to S's X.509 certificate and should contain other fields listed in RFC 7517).

- Signing with a JSON Web Signature provides the protection against another party posing as S by requiring that S actually signs their public key thus proving that they are in possession of the associated private key.

Step 2 E retrieves S's X.509 certificate and verifies it using the certificate chain

- If S's X.509 is signed by CAs recognised by E, and E succesfully validates it using the certificate chain, E proceeds to **Step 3**.
    - If E cannot validate S's X.509 certificate, E sends a reponse to S containing message AUTHENTICATION_FAILURE and error information (reason for failure to validate) and end communication with S.
- If S's X.509 certificate is self-signed, E responds to S with a message AUTHENTICATION_PENDING and request its operator (the operator of ESC E) to validate S's certificate. Successful certificate validation triggers **Step 3**; failure to validate should trigger E sending an AUTHENTICATION_FAILURE message to S.

Step 3 E sends its public key using the JSON Web Key notation with at least the fields listed in **Step 2**, signed with a JSON Web Signature.

Step 4 Mirroring Step 2, S authenticates E by validating E's X.509 associated with its private key and verifies the signature

4.1To do

- define response message format in Step 2
- discuss *2. Exchange of Private Key for digital signature"*. Private keys do not need (should not need!) to be exchanged. Symmetric keys also do not need to be exchanged because JWON Web Encryption has an inbuilt symmetric key mechanism and only needs parties to know each other's public keys.

```
Fallback to TLS 1.1 and TLS 1.0 should be disabled by all parties.
```

```
 Describe Key Bridge Authentication Method
 This page/section should describe how above steps are performed.
```

If the client type is confidential, the client and authorization server establish a client authentication method suitable for the security requirements of the authorization server. The authorization server MAY accept any form of client authentication meeting its security requirements.

Confidential clients are typically issued (or establish) a set of client credentials used for authenticating with the authorization server (e.g., password, public/private key pair).

The client MUST NOT use more than one authentication method in each request.

5Authorization

Authorization is based on OAuth 2.0 Implicit Flow as described in RFC 6749 "The OAuth 2.0 Authorization Framework".

OAuth 2.0 enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.

In OAuth, the client requests access to resources controlled by the resource owner and hosted by the resource server, and is issued a different set of credentials than those of the resource owner. Instead of using the resource owner's credentials to access protected resources, the client obtains an access token -- a string denoting a specific scope, lifetime, and other access attributes. Access tokens are issued to third-party clients by an authorization server with the approval of the resource owner. The client uses the access token to access the protected resources hosted by the resource server.

For example, an end-user (resource owner) can grant a printing service (client) access to her protected photos stored at a photo-sharing service (resource server), without sharing her username and password with the printing service. Instead, she authenticates directly with a server trusted by the photo-sharing service (authorization server), which issues the printing service delegation-specific credentials (access token).

To request an access token, the client obtains authorization from the resource owner. The authorization is expressed in the form of an authorization grant, which the client uses to request the access token.

OAuth defines four grant types:

- authorization code
- implicit
- resource owner password credentials, and
- client credentials.

It also provides an extension mechanism for defining additional grant types.

In this application, implicit grant is used.

5.1Roles

OAuth defines four roles:

5.0.11. Resource Owner (ESC Operator):

> An entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as an end-user.

5.0.22. Resource Server (ESC Server):

> The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens.

5.0.33. Client (SAS):

> An application making protected resource requests on behalf of the resource owner and with its authorization. The term "client" does not imply any particular implementation characteristics (e.g., whether the application executes on a server, a desktop, or other devices).

5.0.44. Authorization Server:

> The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization. The authorization server may be the same server as the resource server or a separate entity.

Additionally, in the context of Implicit flow, user agent is also involved:

5.0.55. User Agent:

> The agent used by the Resource Owner to interact with the Client, for example a browser or a native application.

7Abstract OAuth 2.0 Protocol Flow

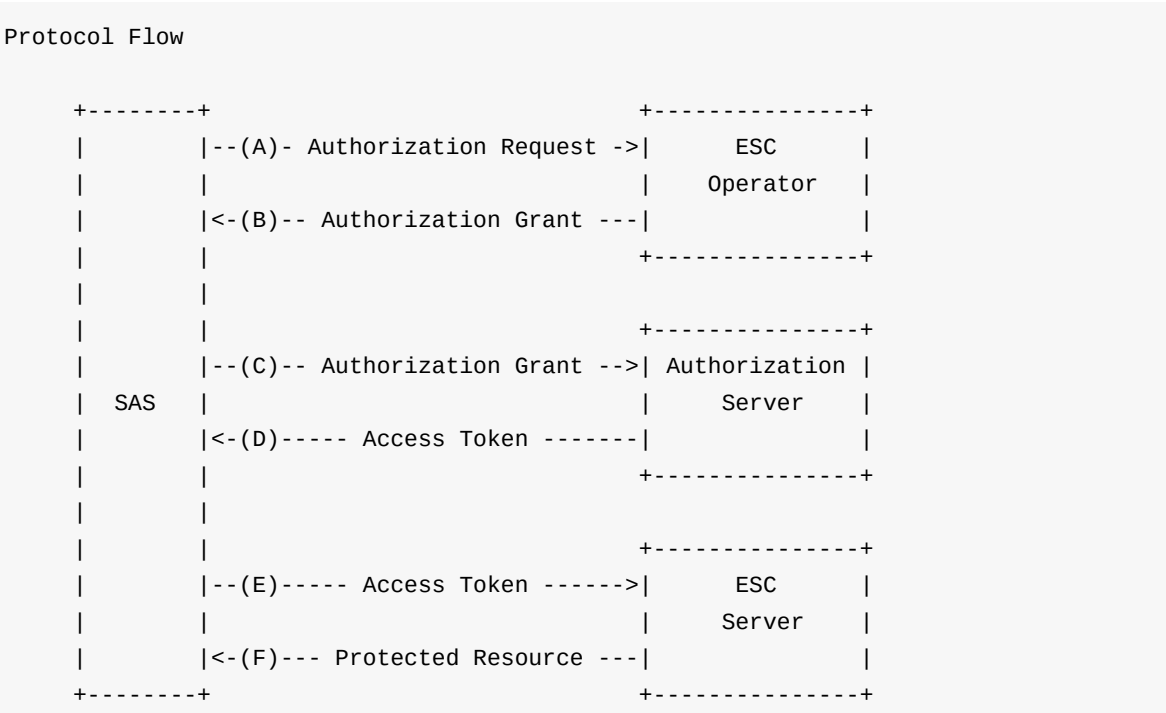


Figure 1: Abstract Protocol Flow

Based on RFC 6749, the abstract OAuth 2.0 flow illustrated in Figure 1 describes the interaction between the four roles and includes the following steps:

- (A) The SAS requests authorization from the ESC Operator. The authorization request can be made directly to the ESC Operator (as shown), or preferably indirectly via the authorization server as an intermediary.
- (B) The SAS receives an authorization grant, which is a credential representing the ESC Operator's authorization, expressed using one of four grant types defined in this specification or using an extension grant type. The authorization grant type depends on the method used by the SAS to request authorization and the types supported by the authorization server.
- (C) The SAS requests an access token by authenticating with the authorization server and presenting the authorization grant.
- (D) The authorization server authenticates the SAS and validates the authorization grant, and if valid, issues an access token.
- (E) The SAS requests the protected resource from the resource server and authenticates by presenting the access token.
- (F) The ESC server validates the access token, and if valid, serves the request.

The preferred method for the SAS to obtain an authorization grant from the ESC Operator (depicted in steps (A) and (B)) is to use the authorization server as an intermediary, which is illustrated in the figure below.

7.1Obtaining Authorization

To request an access token, the client obtains authorization from the resource owner. The authorization is expressed in the form of an authorization grant, which the client uses to request the access token. OAuth defines four grant types:

- authorization code
- implicit
- resource owner password credentials, and
- client credentials
  It also provides an extension mechanism for defining additional grant types.

Key Bridge ESC implements Implicit grant.

7.0.1Authorization Grant

An authorization grant is a credential representing the resource owner's authorization (to access its protected resources) used by the client to obtain an access token. This specification defines four grant types -- authorization code, implicit, resource owner password credentials, and client credentials -- as well as an extensibility mechanism for defining additional types.

7.0.1.1Implicit Authorization Grant

The implicit grant is a simplified authorization code flow optimized for clients implemented in a browser using a scripting language such as JavaScript. In the implicit flow, instead of issuing the client an authorization code, the client is issued an access token directly (as the result of the resource owner authorization). The grant type is implicit, as no intermediate credentials (such as an authorization code) are issued (and later used to obtain an access token).

When issuing an access token during the implicit grant flow, the authorization server does not authenticate the client. In some cases, the client identity can be verified via the redirection URI used to deliver the access token to the client. The access token may be exposed to the resource owner or other applications with access to the resource owner's user-agent.

Implicit grants improve the responsiveness and efficiency of some clients (such as a client implemented as an in-browser application), since it reduces the number of round trips required to obtain an access token.

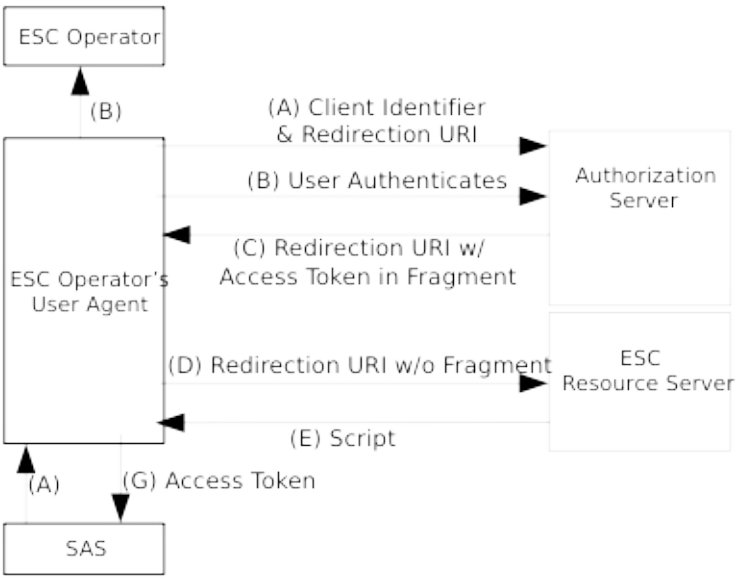
9Implicit Flow - Authorization

The implicit grant type is used to obtain access tokens (it does not support the issuance of refresh tokens) and is optimized for public clients known to operate a particular redirection URI. These clients are typically implemented in a browser using a scripting language such as JavaScript.

Since this is a redirection-based flow, the client must be capable of interacting with the resource owner's user-agent (typically a web browser) and capable of receiving incoming requests (via redirection) from the authorization server.

Unlike the authorization code grant type, in which the client makes separate requests for authorization and for an access token, the client receives the access token as the result of the authorization request.

The implicit grant type does not include client authentication, and relies on the presence of the resource owner and the registration of the redirection URI. Because the access token is encoded into the redirection URI, it may be exposed to the resource owner and other applications residing on the same device.




```
Note: The lines illustrating steps (A) and (B) are broken into two
parts as they pass through the user-agent.

                    Figure 4: Implicit Grant Flow
```

Based on RFC 6749, the flow illustrated in Figure 4 includes the following steps:

(A) The SAS client initiates the flow by directing the ESC Operator's user-agent to the authorization endpoint. The SAS includes its SAS client identifier, requested scope, local state, and a redirection URI to which the authorization server will send the user-agent back once access is granted (or denied).

(B) The authorization server authenticates the ESC Operator (via the user-agent) and establishes whether the ESC Operator grants or denies the client's access request.

(C) Assuming the ESC Operator grants access, the authorization server redirects the user-agent back to the SAS client using the redirection URI provided earlier. The redirection URI includes the access token in the URI fragment.

(D) The user-agent follows the redirection instructions by making a request to the web-hosted client resource (which does not include the fragment per [RFC2616]). The user-agent retains the fragment information locally.

(E) The web-hosted client resource returns a web page (typically an HTML document with an embedded script) capable of accessing the full redirection URI including the fragment retained by the user-agent, and extracting the access token (and other parameters) contained in the fragment.

(F) The user-agent executes the script provided by the web-hosted client resource locally, which extracts the access token.

(G) The user-agent passes the access token to the SAS client.

10Implicit Flow - Authorization Request

The client constructs the request URI by adding the following parameters to the query component of the authorization endpoint URI using the "application/x-www-form-urlencoded" format, per Appendix B:

FIELD NAME	R/O/C	DESCRIPTION
response_type	REQUIRED	Value MUST be set to "token".
client_id	REQUIRED	The client identifier
redirect_uri	OPTIONAL	As described in Section 3.1.2.
scope	OPTIONAL	The scope of the access request as described by Section 3.3
state	RECOMMENDED	An opaque value used by the client to maintain state between the request and callback. The authorization server includes this value when redirecting the user-agent back to the client. The parameter SHOULD be used for preventing cross-site request forgery as described in Section 10.12.

The client directs the resource owner to the constructed URI using an HTTP redirection response, or by other means available to it via the user-agent.

For example, the client directs the user-agent to make the following HTTP request using TLS (with extra line breaks for display purposes only):

```
GET /authorize?response_type=token&client_id=s6BhdRkqt3&state=xyz
    &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
Host: server.example.com
```

The authorization server validates the request to ensure that all required parameters are present and valid. The authorization server MUST verify that the redirection URI to which it will redirect the access token matches a redirection URI registered by the client as described in Section 3.1.2.

If the request is valid, the authorization server authenticates the resource owner and obtains an authorization decision (by asking the resource owner or by establishing approval via other means).

When a decision is established, the authorization server directs the user-agent to the provided client redirection URI using an HTTP redirection response, or by other means available to it via the user-agent.

11Implicit Flow - Access Token Response

If the resource owner grants the access request, the authorization server issues an access token and delivers it to the client by adding the following parameters to the fragment component of the redirection URI using the "application/x-www-form-urlencoded" format, per Appendix B:

FIELD NAME	R/O/C	DESCRIPTION
access_token	REQUIRED	The access token issued by the authorization server.
token_type	REQUIRED	The type of the token issued as described in Section 7.1. Value is case insensitive.
expires_in	RECOMMENDED	The lifetime in seconds of the access token. For example, the value "3600" denotes that the access token will expire in one hour from the time the response was generated. If omitted, the authorization server SHOULD provide the expiration time via other means or document the default value.
scope	OPTIONAL if identical to the scope requested by the client; otherwise, REQUIRED	The scope of the access token as described by Section 3.3.
state	REQUIRED if the "state" parameter was present in the client authorization request.	The exact value received from the client.

The authorization server MUST NOT issue a refresh token.

For example, the authorization server redirects the user-agent by sending the following HTTP response (with extra line breaks for display purposes only):

```
HTTP/1.1 302 Found
Location: http://example.com/cb#access_token=2YotnFZFEjr1zCsicMWpAA
          &state=xyz&token_type=example&expires_in=3600
```

Developers should note that some user-agents do not support the inclusion of a fragment component in the HTTP "Location" response header field. Such clients will require using other methods for redirecting the client than a 3xx redirection response -- for example, returning an HTML page that includes a 'continue' button with an action linked to the redirection URI.

The client MUST ignore unrecognized response parameters. The access token string size is left undefined by this specification. The client should avoid making assumptions about value sizes. The authorization server SHOULD document the size of any value it issues.

12Implicit Flow - Error Response

If the request fails due to a missing, invalid, or mismatching redirection URI, or if the client identifier is missing or invalid, the authorization server SHOULD inform the resource owner of the error and MUST NOT automatically redirect the user-agent to the invalid redirection URI.

If the resource owner denies the access request or if the request fails for reasons other than a missing or invalid redirection URI, the authorization server informs the client by adding the following parameters to the fragment component of the redirection URI using the "application/x-www-form-urlencoded" format, per Appendix B:

FIELD NAME	R/O/C	DESCRIPTION
error	REQUIRED	A single ASCII [USASCII] error code from the Error Codes list below. Values for the "error" parameter MUST NOT include characters outside the set %x20-21 / %x23-5B / %x5D-7E.
error_description	OPTIONAL	Human-readable ASCII [USASCII] text providing additional information, used to assist the client developer in understanding the error that occurred. Values for the "error_description" parameter MUST NOT include characters outside the set %x20-21 / %x23-5B / %x5D-7E.
error_uri	OPTIONAL	A URI identifying a human-readable web page with information about the error, used to provide the client developer with additional information about the error. Values for the "error_uri" parameter MUST conform to the URI-reference syntax and thus MUST NOT include characters outside the set %x21 / %x23-5B / %x5D-7E.
state	REQUIRED	if a "state" parameter was present in the client authorization request. The exact value received from the client.

12.0.0.1Error Codes List

```
invalid_request
      The request is missing a required parameter, includes an
      invalid parameter value, includes a parameter more than
      once, or is otherwise malformed.

unauthorized_client
      The client is not authorized to request an access token
      using this method.

access_denied
      The resource owner or authorization server denied the
      request.

unsupported_response_type
      The authorization server does not support obtaining an
      access token using this method.

invalid_scope
      The requested scope is invalid, unknown, or malformed.

server_error
      The authorization server encountered an unexpected
```

```
      condition that prevented it from fulfilling the request.
      (This error code is needed because a 500 Internal Server
      Error HTTP status code cannot be returned to the client
      via an HTTP redirect.)

temporarily_unavailable
      The authorization server is currently unable to handle
      the request due to a temporary overloading or maintenance
      of the server.  (This error code is needed because a 503
      Service Unavailable HTTP status code cannot be returned
      to the client via an HTTP redirect.)
```

For example, the authorization server redirects the user-agent by sending the following HTTP response:

```
   HTTP/1.1 302 Found
   Location: https://client.example.com/cb#error=access_denied&state=xyz
```

13Message exchange flow and message types

Message exchanges between SAS and ESC are shown in Figures 1 and 2 below.

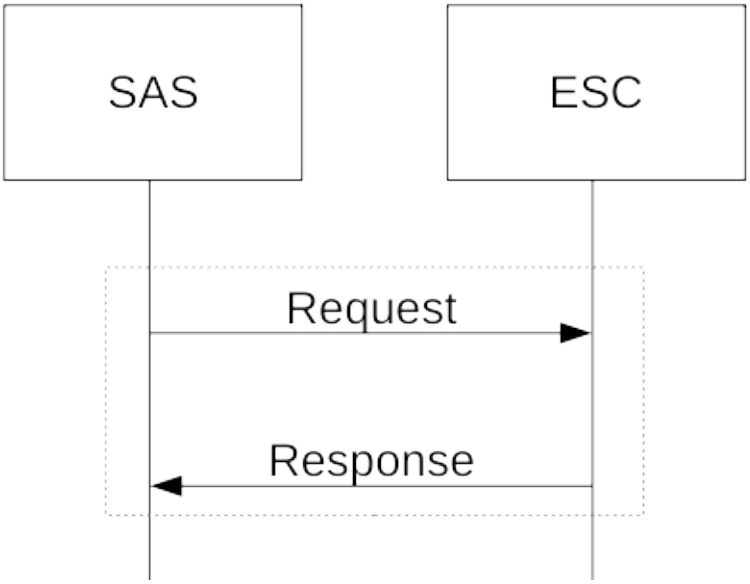


Figure 1: Request-Response flow

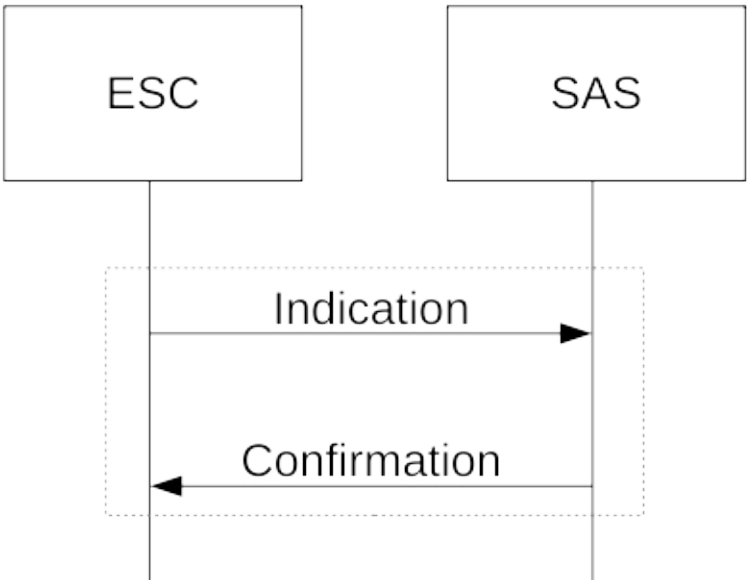


Figure 2: Indication-Confirmation flow

As per this message flow, the following messages shall be exchanged between SAS and ESC.

- **SAS Registration Message**: The message for the SAS to register with the ESC. Request-Response flow in the figure 1 is used.
- **ESC Information Update Message**: The message for the ESC to indicate the update of the ESC information to the SAS. Indication-Confirmation flow in the figure 2 is used.
- **DPA State Message**: The message for the ESC to indicate the DPA activation status to the SAS. Indication-Confirmation flow in the figure 2 is used.

- **Keep Alive Message**: The message for the SAS to detect a failure with ESC. Request-Response flow in the figure 1 is used.
- **SAS Deregistration Message**: The message for the SAS to deregister from the ESC. Request-Response flow in the figure 1 is used.

14Message encoding and transport

14.1Message encoding

The contents of SAS-ESC messages shall be generated by encoding the MessageContainer object using JSON (JavaScript Object Notation) as defined in RFC 7159 [n.2]. Unicode characters shall be used and its default encoding shall be UTF-8.

14.2Message transport

HTTPS shall be used as the transport protocols for SAS-ESC message exchanges. The TLS protocol as specified in [n.3] and HTTP version 1.1 as specified in [n.4] shall be used.

The HTTP POST method shall be used for all requests from the ESC to the SAS and from the SAS to the ESC.

POST shall be sent to the URL provided by the SAS or the ESC. The URL shall be configured in accordance with format `$BASE_URL/$RELEASE_NUMBER/$METHOD_NAME`, where

- `$BASE_URL` represents the base URL of the SAS or the ESC.
- `$RELEASE_NUMBER` represents the CBRS release number corresponding to the ESC Requirements developed by WinnForum [n.1]. In this specification, `$RELEASE_NUMBER` shall be “v1.3”.
- `$METHOD_NAME` represents method name of the API specified in this document and corresponding to the specific SAS-ESC message. The methods in the following table shall be used in the API.

Table 1: List of methods in API

SAS-ESC message name	$METHOD_NAME
SAS Registration Message	sasRegistration
ESC Information Update Message	escUpdate
DPA Activation Status Message	dpaStatusMessage
Keep Alive Message	keepAlive
SAS Deregistration Message	sasDeregistration

Success or failure of any request shall be indicated by using appropriate HTTP status codes.

Table 2: Response status codes

HTTP Status	Meaning	Condition
200	SUCCESS	Request successfully processed
400	BAD REQUEST	Request contains invalid or malformed parameters
404	NOT FOUND	Invalid or malformatted URL

16SAS-ESC Messages

16.1Message Container

All the SAS-ESC Messages shall be generated by encoding the MessageContainer object using JSON. The MessageContainer object is a Flattened JSON Web Signature object as per RFC 7515 § 7.2.2. This representation allows for

- a single signature to digitally sign the payload
- integrity protection (authentication) for the payload and header
- an optional non integrity-protected header

The JWS specification allows for a number of signature algorithms but only ones using public key cryptography should be used.

Table 1: SAS-ESC message signature algorithms

"alg" parameter value	Digital signature algorithm
RS256	RSASSA-PKCS1-v1_5 using SHA-256
RS384	RSASSA-PKCS1-v1_5 using SHA-384
RS512	RSASSA-PKCS1-v1_5 using SHA-512
ES256	ECDSA using P-256 and SHA-256
ES384	ECDSA using P-384 and SHA-384
ES512	ECDSA using P-512 and SHA-512
PS256	RSASSA-PSS using SHA-256 and MGF1 with SHA-256
PS384	RSASSA-PSS using SHA-384 and MGF1 with SHA-384
PS512	RSASSA-PSS using SHA-512 and MGF1 with SHA-512

Table 2: MessageContainer object

Field	Type	R/O/C	Descriptions
protected	string	Required	The value of this parameter is base64url-encoded JOSE header that provides sufficient information to enable the recipient of the message to validate the signature, i.e. the JOSE header object should contain at least the "alg" field with a value from Table 1
header	object JOSEHeader	Optional	An optional JOSEHeader that may be used for transfering additional metadata that do not require integrity protection
payload	string	Required	The value of this field is the base64url-encoded JSON representation of the payload objects listed in section Payload Data
signature	string	Required	This parameter contains the digital signature encoded in Base64URL encoding. This parameter is calculated by taking the BASE64 encoding of the digital signature applied to the Payload Data and Protected header, prepared according to the procedures in Section 3 of RFC 7515 [n.7], using the algorithm as declared in the protected header "alg" field.

Base64 and Base64URL encodings are described in RFC 4648 [n.8].

17.1Payload Data

17.0.1Payload Data for SAS Registration Message

Figure: SAS Registration Request & Response

17.0.1.1SAS Registration Request

SAS Registration Request shall be sent by the SAS to register with the ESC. The SAS Registration Request may be sent also when the SAS information in the *SasRegistrationRequest* object in the following table is updated. The SAS Registration Request shall be generated by encoding the *MessageContainer* object in which JSON-encoded *SasRegistrationRequest* object is used to generate *encodedPayloadData* field using JSON.

Table 3: SasRegistrationRequest object

Field	Type	R/O/C	Descriptions
sasAdministratorId	string	Required	This field shall be included to indicate which SAS Administrator manages the SAS. The format of this field shall be same as $ADMINISTRATOR_ID used in the SAS-SAS Protocol [n.5].
sasImplementationId	string	Required	This field shall be included to indicate the identification of the SAS. The format of this field shall be same as $SAS_IMPLEMENTATION used in the SAS-SAS Protocol [n.5].
publicKey	string	Required	This field shall be included to indicate the public key of SAS.
baseUrl	string	Required	This field shall be included to indicate the base URL ($BASE_URL) of the SAS identified by the *sasImplementationId* field in this object.

17.0.1.2SAS Registration Response

SAS Registration Response shall be sent by the ESC to the SAS for the response to the SAS Registration Request. The SAS Registration Response shall be generated by encoding the *MessageContainer* object in which JSON-encoded *SasRegistrationResponse* object is used to generate *encodedPayloadData* field. in the following table using JSON.

Table 4: SasRegistrationResponse object

Field	Type	R/O/C	Descriptions
sasRegistrationId	string	Required	This field shall be generated by the ESC and included to indicate the registration identifier for the SAS.

escOperatorId	string	Required	This field shall be included to indicate the identification of ESC Operator managing ESC. The format of this field shall be FFS.
escInformation	object EscInformation	Required	This field shall be included to indicate the ESC information.

Table 5: EscInformation object

Field	Type	R/O/C	Descriptions
escImplementationId	string	Required	This field shall be included to indicate the identification of the ESC implementation. The format of this field shall be FFS.
escSensors	array of object EscSensorData[5]	Required	This field shall be included to indicate the information of ESC sensors deployed by the ESC Operator and managed by the ESC identified by the *escOperatorId* and *escImplementationId* fields in this object, respectively. See details in 7.1.2.1.

`Editor's Note: Should the format of each ID be "ESC-CA certified unique ESC Operator identifier" similar to both SAS Administrator and Implementation ID?`

`Editor's Note: "escImplementationId" is included here for the similar purpose of "SAS Implementation" in SAS-SAS. In other words, one or more ESC instances might be operated. Decision to remove or keep this field strongly depends on Key Bridge for now.`

18.0.0.0.1Enhancements to EscSensorData object for SAS-ESC Interface

In this specification, the EscSensorData object specified in the SAS-SAS Protocol [n.5] shall be reused with enhancements. Enhanced definition of the EscSensorData object is described in the following table.

Table 6: Enhanced definition of EscSensorData object

Field	Type	R/O/C	Descriptions
id	string	N/A	This field shall not be included.
sensorId	string	Required	This field shall be included to indicate a unique identifier of the ESC Sensor.
installationParam	object InstallationParam	N/A	This field shall not be included.
escInstallationParam	object EscInstallationParam	Required	This field shall be included to indicate the installation parameters of the ESC Sensor identified by thesensorIdfield in this object.
protectionLevel	number	Required	This field shall be included to indicate the protection level to be applied to the ESC Sensor identified by thesensorIdfield in this object. The value of this field shall be in units of dBm/MHz with decimal point.

Table 7: EscInstallationParam object

Field	Type	R/O/C	Descriptions
latitude	number	Required	Latitude of the ESC Sensor location in degrees relative to the WGS 84 datum. The allowed range is from -90.000000 to +90.000000. Positive values represent latitudes north of the equator; negative values south of the equator. Values are specified using 6 digits to the right of the decimal point.
longitude	number	Required	Longitude of the ESC Sensor location in degrees relative to the WGS84 datum. The allowed range is from -180.000000 to +180.000000. Positive values represent longitudes east of the prime meridian; negative values west of the prime meridian. Values are specified using 6 digits to the right of the decimal point.
height	number	Required	The antenna height of ESC Sensor in meters. When the heightType parameter value is “AGL”, the antenna height shall be given relative to ground level. When the heightType parameter value is “AMSL”, it is given with respect to WGS84 datum.
heightType	string	Required	The value shall be “AGL” or “AMSL”. AGL height is measured relative to the ground level. AMSL height is measured relative to the mean sea level.
horizontalAccuracy	number	Required	A positive number in meters to indicate accuracy oftheESC Sensorantennahorizontal location.
verticalAccuracy	number	Required	A positive number in meters to indicate accuracy oftheESC Sensorvertical location.
antennaAzimuth	number	Required	Boresight direction of the horizontal plane of the antenna in degrees with respect to true north. The value of this parameter is an integer with a value between 0 and 359 inclusive. A value of 0 degrees means true north; a value of 90 degrees means east.
			Antenna down tilt in degrees and is an integer with a value

antennaDowntilt	number	Required	between -90 and +90 inclusive; a negative value means the antenna is tilted up (above horizontal).
azimuthAntennaPattern	array of object AntennaPattern	Required	This parameter specifies anantenna pattern in any direction for the ESC Sensor antenna in the azimuthal plane.

Table 8: AntennaPattern object

Field	Type	R/O/C	Description
angle	number	Required	This is the angle.**In the azimuth plane:**the value is given in degrees relative to the boresight of the antenna. The value of this parameter is an integer between 0 and 360 inclusive.**In the elevation plane**: the angle is given in degrees relative to the horizon. The value of this parameter is an integer between -180 and 180 inclusive.
gain	number	Required	The gain in dBi includes both antenna gain and beamforming gain. This parameter is an integer with a value between -127 and +128 (dBi). The gain provided is the gain in the direction of ‘angle’.

19.0.1Payload Data for ESC Information Update Message

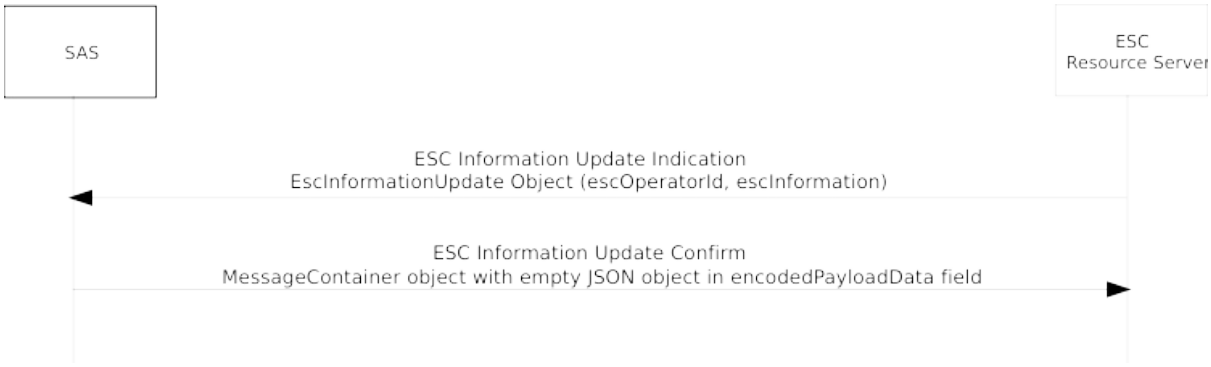


Figure: ESC Information Update Indication & Confirm

19.0.0.1ESC Information Update Indication

ESC Information Update Indication shall be sent by the ESC to the SAS when the ESC information is updated. The ESC Information Update Indication shall be generated by encoding the MessageContainer object in which JSON-encoded EscInformationUpdate object is used to generate encodedPayloadData field in the following table using JSON.

Table 9: EscInformationUpdate object

Field	Type	R/O/C	Description
escOperatorId	string	Required	This field shall be included to indicate the identification of ESC Operator managing ESC. The format of this field shall be FFS.
escInformation	object EscInformation	Required	This field shall be included to indicate the ESC information.

19.0.0.2ESC Information Update Confirm

ESC Information Update Confirm shall be sent by the ESC to the SAS for the response to the ESC Information Update Indication. The ESC Information Update Confirm shall be generated by encoding the *MessageContainer* object in which empty JSON object (i.e. “{}”) is used to generate *encodedPayloadData* field.

20.0.1Payload Data for DPA Activation Status Message

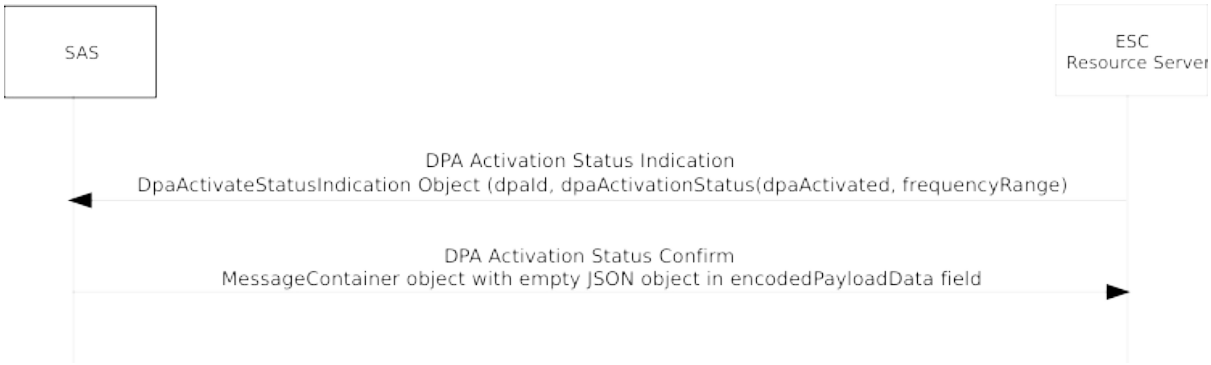


Figure: DPA Activation Status Indication & Confirm

20.0.1DPA Activation Status Indication

DPA Activation Status Indication shall be sent by the ESC to the SAS. The DPA Activation Status Indication shall be generated by encoding the *MessageContainer* object in which the *DpaActivateStatusIndication* object is used to generate *encodedPayloadData* field in the following table using JSON.

Table 10: DpaActivateStatusIndication object

Field	Type	R/O/C	Descriptions
dpaId	string	Required	This field shall be included to indicate which a unique identifier of the DPA.
dpaActivationStatus	object DpaActivationStatus	Required	This field shall be included to indicate the DPA activation status.

Table 11: DpaActivationStatus object

Field	Type	R/O/C	Descriptions
dpaActivated	boolean	Required	This field shall be included to indicate the DPA activation status of the frequency range indicated by the *frequencyRange* field in this object “true”: DPA is (has been) activated “false”: DPA is (has been) deactivated
frequencyRange	object FrequencyRange[6]	Required	This field shall be included to indicate the frequency range.

Editor’s Note: Atomic Transaction = 1 DPA + 1 Channel

Editor’s Note: Open Issues.

1: ID Assignment,

2: Partial coverages (geometry and channel block).

20.0.0.1DPA Activation Status Confirm

DPA Activation Status Confirm shall be sent by the ESC to the SAS for the response to the DPA Activation Status Indication. The DPA Activation Status Confirm shall be generated by encoding the *MessageContainer* object in which empty JSON object (i.e. “{}”) is used to generate *encodedPayloadData* field.

21.0.1Payload Data for Keep Alive Message

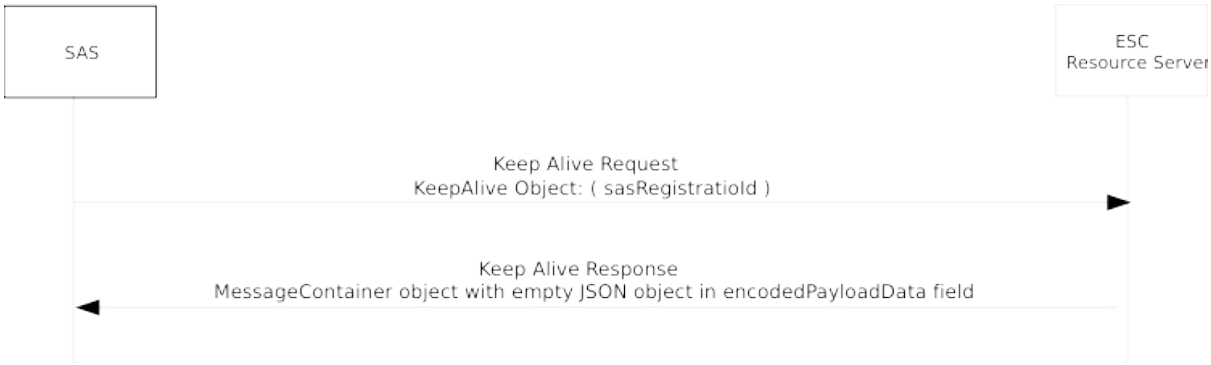


Figure: Keep Alive Indication & Confirm

21.0.0.1Keep Alive Indication

Keep Alive Indication shall be sent from the SAS to ESC. The Keep Alive Request shall be generated by encoding the MessageContainer object in which the *KeepAlive* object is used to generate encodedPayloadData field.in the following table using JSON.

Table 12: KeepAlive object

Field	Type	R/O/C	Descriptions
sasRegistrationId	string	Required	This field shall be included to indicate the registration identifier of the SAS.

21.0.0.2Keep Alive Confirm

Keep Alive Confirm shall be sent from the ESC to the SAS for the response to the Keep Alive Request. The Keep Alive Response shall be generated by encoding the MessageContainer object in which empty JSON object (i.e. “{}”) is used to generate encodedPayloadData field.

22.0.1Payload Data for SAS Deregistration Message

Figure: SAS Deregistration Request & Response

22.0.0.1SAS Deregistration Request

SAS Deregistration Request may be sent by the SAS to deregister from the ESC when the SAS wants to stop receiving indications from the ESC. The SAS Deregistration Request shall be generated by encoding the MessageContainer object in which the SasDeregistrationRequest object is used to generate encodedPayloadData field using JSON.

Table 13: SasDeregistrationRequest object

Field	Type	R/O/C	Descriptions
sasAdministratorId	string	Required	This field shall be included to indicate which SAS Administrator manages the SAS. The format of this field shall be same as $ADMINISTRATOR_ID used in the SAS-SAS Protocol [n.5].
sasRegistrationId	string	Required	This field shall be generated by the ESC and included to indicate the registration identifier for the SAS.

22.0.0.2SAS Deregistration Response

SAS Deregistration Response shall be sent by the ESC to the SAS for the response to the SAS Deregistration Request. The SAS Deregistration Response shall be generated by encoding the MessageContainer object in which empty JSON object (i.e. “{}”) is used to generate encoded PayloadData field.

24Annex A (Normative) DPA State Machine

SAS and ESC employing the SAS-ESC Protocol specified in this present document shall consider the DPA State Machine. This DPA State Machine shall be considered in each channel per DPA. The figure below shows the DPA State Machine in a channel per DPA.

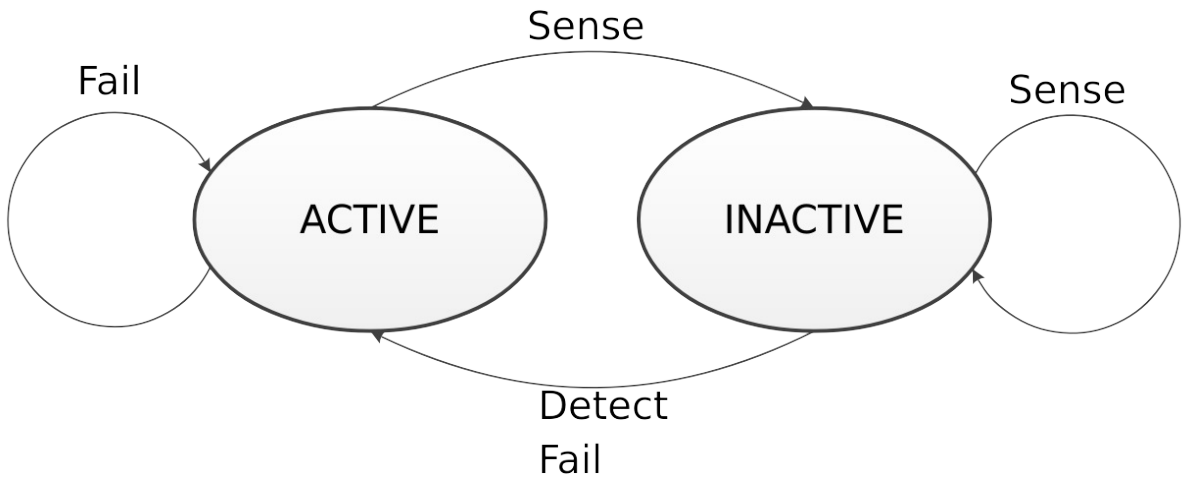


Figure 2: DPA State Machine in a channel per DPA

The DPA State shall be defined as follows:

- ACTIVE: “ACTIVE” refers to the state of a DPA in which the ESC detects the presence of the incumbent, in which two hours haves not passed after the ESC senses disappearance of the presence of any incumbents, or in which CBSDs are forbidden to use .a channel in its Neighborhood Area regardless of the presence of the incumbents. Both SAS and ESC shall consider “ACTIVE” the initial state of a DPA.
- INACTIVE: “INACTIVE” refers to the state of a DPA in which CBSDs are allowed to use. a channel in its Neighborhood Area, in which two hours have passed after the ESC sensed disappearance of the presence of any incumbents when the DPA was ACTIVE State, or in which the ESC senses no presence of the incumbents.

The following trigger events shall be applied to the DPA State Transitions:

- Sense: “Sense” refers to an event in which the ESC senses no presence of the incumbents or in which two hours has passed after the ESC sensed disappearance of the presence of the incumbent. All the DPA States shall transition to “INACTIVE” after this trigger event.
- Detect: “Detect” refers to an event in which the ESC detects the presence of the incumbent in the DPA. All the DPA States shall transition to “ACTIVE” after this trigger event.
- Fail: “Fail” refers to an event in which the SAS detects any failures in the Keep Alive Message exchange with the ESC. In particular, if the SAS doesn’t receive any Keep Alive Response for more than x seconds after sending a Keep Alive Message, it shall move to the “Fail” state. All the DPA States shall transition to “ACTIVE” after this trigger event.

24.1Security Concerns

24.0.1Implicit Grant

Implicit grants improve the responsiveness and efficiency of some clients (such as a client implemented as an in-browser application), since it reduces the number of round trips required to obtain an access token. However, this convenience should be weighed against the security implications of using implicit grants, such as those described in Sections 10.3 and 10.16 of RFC 6749, especially when the authorization code grant type is available.

24.0.2TLS Version

Whenever Transport Layer Security (TLS) is used by this specification, the appropriate version (or versions) of TLS will vary over time, based on the widespread deployment and known security vulnerabilities. At the time of this writing, TLS version 1.2 [RFC5246] is the most recent version, but has a very limited deployment base and might not be readily available for implementation. TLS version 1.0 [RFC2246] is the most widely deployed version and will provide the broadest interoperability.

Implementations MAY also support additional transport-layer security mechanisms that meet their security requirements.

24.0.3Interoperability

OAuth 2.0 provides a rich authorization framework with well-defined security properties. However, as a rich and highly extensible framework with many optional components, on its own, this specification is likely to produce a wide range of non-interoperable implementations.

In addition, this specification leaves a few required components partially or fully undefined (e.g., client registration, authorization server capabilities, endpoint discovery). Without these components, clients must be manually and specifically configured against a specific authorization server and resource server in order to interoperate.

This framework was designed with the clear expectation that future work will define prescriptive profiles and extensions necessary to achieve full web-scale interoperability.

26Abbreviations

For the purposes of the present document, the following abbreviations apply:

Abbreviation	Full Form
API	Application Programming Interface
CBRS	Citizens Broadband Radio Service
CONOPS	Concept of Operations
DPA	Dynamic Protection Area
EIRP	Effective Isotropic Radiated Power
ESC	Environmental Sensing Capability
FFS	For Further Study
FCC	Federal Communications Commission
JSON	JavaScript Object Notation
HTTP	Hyper Text Transfer Protocol
HRRPS	HTTP plus TLS Protocol
NIIU	Non-Informing Incumbent User; information must be learned
RAT	Radio Access Technology
SAS	Spectrum Access System
TLS	Transfer Layer Security
URL	Uniform Resource Locator
WinnForum	Wireless Innovation Forum

27Definitions

27.1Citizens Broadband Radio Service (CBRS)

Wireless operations authorized by the U.S. Federal Communications Commission (FCC) in the 3,550-3,700 MHz frequency band. The CBRS includes Priority Access and General Authorized Access tiers of service.

27.2Environmental Sensing Capability (ESC)

A system that detects and communicates the presence of a signal from an Incumbent User to an SAS to facilitate shared spectrum access consistent with 47 C.F.R. Part 96.

27.3ESC Operator

A legal entity that the FCC has authorized to operate an ESC.

27.4Incumbent User

A federal entity authorized to operate on a primary basis in accordance with the table of frequency allocations, a fixed satellite service operator, or a Grandfathered Wireless Broadband Licensee authorized to operate on a primary basis on frequencies designated in section 96.11.

27.5Protection

To avoid harmful interference from lower-tier user(s) to the upper tier user(s).

27.6SAS Administrator

A legal entity that the FCC has authorized to administer the operation of a SAS.

27.7Spectrum Access System (SAS)

A system that authorizes and manages the use of spectrum for the CBRS in accordance with subpart F of 47 C.F.R. Part 96.

28References

28.1WINNF-TS-0016

[n.6] “WINNF-TS-0016 Signaling Protocols and Procedures for Citizens Broadband Radio Service (CBRS): Spectrum Access System (SAS) - Citizens Broadband Radio Service Device (CBSD) Interface Technical Specification v1.1.0”, Wireless Innovation Forum, 18 July 2017

28.2WINNF-TS-0065

[n.3] “WINNF-TS-0065 CBRS Communications Security Technical Specification v1.1.0”, Wireless Innovation Forum, 26 July 2017

28.3SAS-SAS Protocol

[n.5] “WINNF-TS-0096 Signaling Protocols and Procedures for Citizens Broadband Radio Service (CBRS): Spectrum Access System (SAS) - SAS Interface Technical Specification v1.1.0”, Wireless Innovation Forum, 1 August 2017
https://workspace.winnforum.org/higherlogic/ws/public/document?document_id=4517&wg_abbrev=SSC

28.4WINNF-TS-0112-V1.3.0

[n.0] “WINNF-TS-0112 Requirements for Commercial Operation in the U.S. 3550-3700 MHz Citizens Broadband Radio Service Band v1.3.0”, Wireless Innovation Forum, 27 September 2017 https://workspace.winnforum.org/higherlogic/ws/public/document?document_id=4743

28.5RFC 4648

[n.8] The Base16, Base32, and Base64 Data Encodings
https://tools.ietf.org/html/rfc4648

28.6RFC 6749

OAuth 2.0 for automated identification and token rotation
https://tools.ietf.org/html/rfc6749

28.7RFC 7159

The JavaScript Object Notation (JSON) Data Interchange Format
https://tools.ietf.org/html/rfc7159

28.8RFC 7515

[n.7] JSON Web Signature (JWS)
https://tools.ietf.org/html/rfc7515

28.9RFC 7516

JSON web encryption
https://tools.ietf.org/html/rfc7516

28.10RFC 7517

JSON web key
https://tools.ietf.org/html/rfc7517

28.11RFC 7518

JSON web algorithm
https://tools.ietf.org/html/rfc7518

28.12RFC 7519

JSON web token
https://tools.ietf.org/html/rfc7519

28.13RFC 7797

JWS for unencoded payload
https://tools.ietf.org/html/rfc7797