Workgroup: Network Working Group
Internet-Draft: draft-xu-savax-protocol-03
Published: 21 November 2022
Intended Status: Informational
Expires: 25 May 2023
Authors: K. Xu, Tsinghua University; J. Wu, Tsinghua University; X. Wang, Tsinghua University; Y. Guo, Tsinghua University

Communication Protocol Between the AD Control Server and the AD Edge Router of Inter-Domain Source Address Validation Architecture

Abstract

Because the Internet forwards packets according to the IP destination address, packet forwarding typically takes place without inspection of the source address and malicious attacks have been launched using spoofed source addresses. The inter-domain source address validation architecture is an effort to enhance the Internet by using state machine to generate consistent tags. When communicating between two end hosts at different ADs of IPv6 network, tags will be added to the packets to identify the authenticity of the IPv6 source address.

This memo focus on the packet formats and processing flow of the SAVA-X mechanism.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Inter-Domain Source Address Validation Architecture (SAVA-X) mechanism establishes a trust alliance among Address Domains (AD), maintains a one-to-one state machine among ADs, generates a consistent tag, and deploys the tag to the ADs' border router (AER). The AER of the source AD adds a tag to identify the identity of the AD to the packet originating from one AD and sinking in another AD. The AER of the destination AD verifies the source address by validating the correctness of the tag to determine whether it is a packet with a forged source address.

In the process of packet forwarding, if the source address and the destination address of this packet both are addresses in the trust alliance, however the tag is not added or incorrectly added, AER of the destination AD determines that the source address is forged and directly discards this packet. The destination AD forwards the packet directly for packets whose source address is an address outside the trust alliance.

This document mainly studies the relevant specifications of the data plane of the inter-domain source address validation architecture mechanism between ADs, which will protect IPv6 networks from being forged source address. You could see [RFC8200] for more details about IPv6. It stipulates the state machine, tag generation and update, tag processing in AER, and packet signature Its promotion and application can realize the standardization of the data plane in the SAVA-X to facilitate the related equipment developed by different manufacturers and organizations to cooperate to accomplish the inter-domain source address validation jointly.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119, BCP 14 [RFC2119] and indicate requirement levels for compliant CoAP implementations.

2. Terminology and Abbreviation

Abbreviation	**Description**
AD	Address Domain, the unit of a trust alliance, which is an address set consisting of all IPv6 addresses corresponding to an IPv6 address prefix.
TA	Trust Alliance, the IPv6 network that uses the SAVA-X mechanism.
ACS	AD Control Server, the server that matains state machine with other ACS and distribute information to AER.
AER	

Abbreviation	Description
	AD border router, which is placed at the boundary of an AD of STA.
ADID	The identity of an AD.
ADID_Rec	The record of a number of an AD.
ARI_Rec	The record with relavent information of an AD or STA.
API_Rec	The record of prefix of an AD or STA.
SM	State Machine, which is maintained by a pair of ACS to generate tags.
SMI_Rec	The record of the state machine information.
Tag	The authentic identification of source address of a packet.

Table 1

3. Communication Protocol Format

Every AD should be placed at least one ACS, which is mainly responsible for maintaining the relationship between ADs of the trust alliance, establishing connections with other ACS, maintaining the synchronous state machine, and sending the generated tags to the AER. TCP is used for communicating between ACS-ACS and ACS-AER.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    Version    |    Alliance   | I Type| S Type|   Operation   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                           Total Length                        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                        Number of Records                      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                       Transaction Number                      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                     Acknowledgement Number                    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ~                              Data                             ~
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Version: 8-bit, the current version=0b1 of SAVA-X.

Alliance: 8-bit, the sub-trust alliance number.

I Type: 4-bit, Information type, 0 for G_REF_INFO, 1 for AD_REG_INFO, 2 for AD_PREFIX_INFO, 3 for STATE_MACHINE_INFO, 4 for DIAGNOSIS_INFO, 5 for RUNNING_STATE_INFO, 6 for

STRATEGY_INFO, 7 for ALIVE_INFO, 8 for TAG_INFO, 9 for ALLI_TAG_INFO, 10 for AD_V_TAG_INFO and others are unassigned.

S Type: 4-bit, Session type, 1 for ANNOUNCEMENT or DEPLOYMENT, 2 for REQUEST, 3 for REQUEST_ALL, 4 for ACK, 5 for NAK, 6 for AACK, 7 for ANAK, 8 for RACK, 9 for RNAK and others are unassigned.

Operation: 8-bit, the first 3 bits means for whether RENEW Type or not. First bit: 0 for non-RENEW packet, 1 for RENEW packet. Second bit: 0 for the first non-RENEW packet, 1 for the first RENEW packet. Third bit: 0 for the last non-RENEW packet, 1 for the last RENEW packet.

Total Length: 32-bit, the length of this packet: from Version to Data.

Number of Records: 32-bit, he records in Data.

Transaction Number: 32-bit, this is the identification of a publication, query or response, and the value should increase monotonically. And different I Types MUST have its own Transaction Number. Through this field, ACS can locate which information has been resolved wrongly and corrected it.

Acknowledgement Number: 32-bit, it is only be filled when S Type is ACK, NAK, AACK, ANAK, RACK or RNAK. Otherwise it is should be filled as 0.

Data: Variable-length field. I Type and S Type specifies data jointly.

When S Type is ANNOUNCEMENT:

*If I Type = AD_REG_INFO, Data field SHOULD be one or more ARI_Rec.

*If I Type = AD_PREFIX_INFO, Data field SHOULD be one or more API_Rec.

*If I Type = STATE_MACHINE_INFO, Data field SHOULD be one or more SMI_Rec.

*If I Type = TAG_INFO, ALLI_TAG_INFO or AD_V_TAG_INFO, Data field SHOULD be one or more TAG_Rec.

When S Type is REQUEST or REQUEST_ALL:

*If I Type = REG_INFO, Data field SHOULD be one or more ADID_Rec.

* If I Type = AD_PREFIX_INFO, Data field SHOULD be none or one or more ADID_Rec.

* If I Type = STATE_MACHINE_INFO, Data field SHOULD be none or one or more ADID_Rec.

* If I Type = DIAGNOSE_INFO, Data field SHOULD be a 32-bit diagnose reqeust code.

* If I Type = ALIVE_INFO, Data field SHOULD be none.

When S Type is ACK, AACK or RACK:

* If I Type = REG_INFO, Data field SHOULD be one or more ARI_Rec.

* If I Type = AD_PREFIX_INFO, Data field SHOULD be one or more API_Rec.

* If I Type = STATE_MACHINE_INFO, Data field SHOULD be one or more SMI_Rec.

* If I Type = DIAGNOSE_INFO, Data field SHOULD be one 32-bit diagnose response code.

* If I Type = ALIVE_INFO, Data field SHOULD be none.

When S Type is NAK, ANAK or RNAK, Data field SHOULD be one 32-bit error code:

* 1 for parameters are wrong which means the packet cannot resolve correctly.

* 2 for member AD(s) in request packet does not exist in the designative sub-trust alliance.

* 3 for algorithm for State Machine set by source ACS cannot support by the destination ACS.

4. ACS-ACS Communication Protocol

Since the blockchain is adopted in SAVA-X to maintain the information of the trust alliance, ACS can query the address domain information of relevant ADes of the trust alliance and the AD prefix information corresponding to the address domain from the blockchain.

4.1. Announcement, Query and Response of State Machine Information

State machine information record (SMI_Rec) represents the packet format used when state machine is negotiated between different ordered pairs of ADs. When an ordered pair of ADs is negotiating the state machine, ACS of AD with smaller ADID initiates the communication and ACS of AD with larger ADID uses SMI_Rec determines the information to be used, such as initial state, tag generation algorithm, state transition interval, etc. Compared to ARI_Rec and API_Rec,SMI_Rec also needs an Expiring Time in addition to the Effecting Time. Expiration Time stands when the negotiated state machine is no longer valid.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|     Action    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source ADID_Rec                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Destination ADID_Rec                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       State Mathine ID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Algorithm              |              IS Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                        Initial State                          ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Transition Interval                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Effecting Time                         |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Expiring Time                          |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Action: 8-bit, 1 for add or update this SMI_Rec.

Source ADID_Rec: Variable-length field. Refer to ADID_Rec [SAVA-X-Control].

Destination ADID_Rec: Variable-length field. Refer to ADID_Rec in [SAVA-X-Control].

State Mathine ID: 32-bit, the ID used to identify the state machine, which is unique to a specific ordered AD pair and grows monotonically in use. It is used to distinguish the sequence before and after the generation of multiple state machines.

Algorithm: 16-bit, algorithm used in A-Box. 1 for KISS-99 32-bit, 2 for KISS-99 64-bit Joint, 3 for OTP-2289 MD5 and others are unassigned.

IS Length:
16-bit, the length of Initial State field.

Initial State: Variable-length field, the length of this filed is determined by IS Length.

Transition Interval: 32-bit, the milliseconds of interval of state transition.

Effecting Time: 64-bit, when this field is 0, it means this State Machine should be enabled after the last State Machine expired.

Expiring Time: 64-bit, the end of this State Machine.

4.1.1. State Machine Information Announcement

State machine information announcement (SM_INFO-Announce) is sent from source ACS to destination ACS. Source ACS fills in the following values for each field:

Field	**Value**
Version	1
Alliance	The sub-trust alliance number.
I Type	SM_INFO
S Type	ANNOUNCEMENT
Operation	NULL: source ACS updates part of the state machines information to destination ACS. RENEW: source ACS updates all the state machines information to destination ACS.
Total Length	The length of this message.
Number of Records	The number of SMI_Recs in Data field.
Transaction Number	The last Transaction Number add 1. ACS would maintain a global Transaction Number for packets sent out where I Type is SM_INFO and ACS would keep it increasing monotonic.
Acknowledgement Number	0
Data	One or more SMI_Recs.

Table 2

All SMI_Recs in Data field should have an unique SM_ID. When Action is ADD and SM_ID bigger than current used SM_ID, ACS should add the state machine defined in SMI_Rec. When Action is ADD and SM_ID equals to current used SM_ID, ACS should modify the state machine defined in SMI_Rec. Only Transition Interval and Expiring Time can be modified. Other SMI_Rec should be discarded and destination ACS should send a NAK message to source ACS.

When receiving a non-RENEW packet, if it cannot resolve this message, destination ACS should send a NAK message to source ACS. When destination ACS can resolve the packet correctly, it SHOULD:

1. Compare the Transaction Number in this packet with Transaction Number received from the same ACS before. Otherwise, destination ACS would discard this packet and send a SM_INFO-Request to request the lastest information of state machine. SM_INFO-Request is defined at Section 4.1.2. If bigger, destination ACS WOULD:

2. Accept every SMI_Rec and process them as following: - If the SM_ID in SMI_Rec equals to current used SM_ID, destination ACS would update the current used SM_ID. - If the SM_ID in SMI_Rec bigger than current used SM_ID, destination ACS would add this state machine to its following used state machine list.

3. And then destination ACS will send an SM_INFO-AACK message to source ACS.

When receiving a RENEW packet, if it cannot resolve this message, destination ACS should send a SM_INFO-ANAK message to source ACS. When destination ACS can resolve the packet correctly, it SHOULD:

1. Compare the Transaction Number in this packet with Transaction Number received from the same ACS before. Otherwise, destination ACS would discard this packet and send a SM_INFO-Request to request the lastest information of state machine. If bigger, destination ACS WOULD:

2. Accept every SMI_Rec and process them as following: - If the SM_ID in SMI_Rec equals to current used SM_ID, destination ACS would update the current used SM_ID. - If the SM_ID in SMI_Rec bigger than current used SM_ID, destination ACS would add this state machine to its following used state machine list. Especially, state machines will be removed right now when they are not listed in the SMI_Recs but they are in using.

3. And then destination ACS will send an SM_INFO-AACK message to source ACS.

There are two types of reply of SM_INFO-Annouce message. That is SM_INFO-AACK representing affirmative acknowledgement and SM_INFO-ANAK representing negative acknowledgement. These are sent from destination ACS to source ACS. The mainly part of packet is filled by destination ACS as follows:

Field	Value
Version	1

Field	**Value**
Alliance	The sub-trust alliance number.
I Type	SM_INFO
S Type	AACK if it is affirmative acknowledgement or ANAK if it is negative acknowledgement.
Operation	NULL
Total Length	The length of this message.
Number of Records	0
Transaction Number	The last Transaction Number add 1. ACS would maintain a global Transaction Number for packets sent out where I Type is SM_INFO and ACS would keep it increasing monotonic.
Acknowledgement Number	The Transaction Number of the response corresponding request.
Data	S Type = AACK: None. S Type = ANAK: a 32-bit error code defined in Section 3.

Table 3

Nothing needs to do when source ACS receives an SM_INFO-AACK message while it should regenerate a new state machine and announces to destination ACS when source ACS receives an SM_INFO-ANAK message.

4.1.2. State Machine Information Request

State machine information request (SM_INFO-Request) is sent from source ACS to destination ACS. Source ACS fills in the following values for each field:

Field	**Value**
Version	1
Alliance	The sub-trust alliance number.
I Type	SM_INFO
S Type	REQUEST
Operation	NULL: announce all state machine information to source ACS.
Total Length	The length of this message.
Number of Records	0
Transaction Number	The last Transaction Number add 1. ACS would maintain a global Transaction Number for packets sent out where I Type is SM_INFO and ACS would keep it increasing monotonic.
Acknowledgement Number	0
Data	None

Table 4

When source ACS receives a SM_INFO-Request message, it would send a SM_INFO-RNAK message to destination ACS if some fields are wrong. Otherwise, source ACS would send an SM_INFO-RACK message to destination ACS and process this SM_INFO-Request message. Source ACS should compare the Transaction Number in this message with Transaction Number received from the same destination ACS before. Otherwise, source ACS would discard this packet. If bigger, source ACS would send an SM_INFO-RACK message to destination ACS.

There are two types of reply of SM_INFO-Request message, i.e. SM_INFO-RACK representing affirmative acknowledgement and SM_INFO-RNAK representing negative acknowledgement. These are sent from source ACS to destination ACS. The mainly part of packet is filled by source ACS as follows: I Type is SM_INFO. S Type is RACK if it is affirmative acknowledgement or RNAK if it is negative acknowledgement. Operation is NULL. When S Type is RACK, Data field is a few of SMI_Recs. When S Type is RNAK, Data field is a 32-bit error code.

When receiving a SM_INFO-RACK message, if it cannot resolve this message, destination ACS should send a SM_INFO-Request message to source ACS to acquire another state machine. When destination ACS can resolve the message correctly, it SHOULD:

1. Compare the Transaction Number in this packet with Transaction Number received from the same source ACS before. Otherwise, destination ACS would discard this packet and send a SM_INFO-Request to request the lastest information of state machine. If bigger, destination ACS WOULD:

2. Accept every SMI_Rec and process them as following: - If the SM_ID in SMI_Rec equals to current used SM_ID, destination ACS would update the current used SM_ID. - If the SM_ID in SMI_Rec bigger than current used SM_ID, destination ACS would add this state machine to its following used state machine list.

3. And then destination ACS will send an SM_INFO-AACK message to source ACS.

When receiving a SM_INFO-RNAK message, if it cannot resolve this message, destination ACS should send a SM_INFO-Request message to source ACS to acquire new state machine. When destination ACS can resolve the message correctly, it SHOULD compare the Transaction Number in this packet with Transaction Number received from the same source ACS before. Otherwise, destination ACS would discard this packet and send a SM_INFO-Request to request the lastest information of state machine. If bigger, destination ACS WOULD send a new correct SM_INFO-Request message to source ACS.

4.2. Request and Response of Diagnose Information

Sent by destination ACS, request of diagnose information (DIAG_INFO-Request) is used to require the source ACS to check its configuration and source AERs' settings. Source ACS will response with its result. Destination ACS fills in the following values for each field:

Field	Value
Version	1
Alliance	The sub-trust alliance number.
I Type	DIAG_INFO
S Type	REQUEST
Operation	NULL
Total Length	The length of this message.
Number of Records	0
Transaction Number	The last Transaction Number add 1. ACS would maintain a global Transaction Number for packets sent out where I Type is DIAG_INFO and ACS would keep it increasing monotonic.
Acknowledgement Number	0
Data	a 32-bit error code defined below.

Table 5

Response of diagnose information (DIAG_INFO-Response) replys from source ACS to destination ACS.

Field	Value
Version	1
Alliance	The sub-trust alliance number.
I Type	DIAG_INFO
S Type	ACK
Operation	NULL
Total Length	The length of this message.
Number of Records	0
Transaction Number	The last Transaction Number add 1. ACS would maintain a global Transaction Number for packets sent out where I Type is DIAG_INFO and ACS would keep it increasing monotonic.
Acknowledgement Number	The Transaction Number of the response corresponding request.
Data	a 32-bit error code defined below.

Table 6

Before it sends the DIAG_INFO-Request message, the destination ACS should check its own configuration and gurantee they are correct.

If it receives a DIAG_INFO-Request message, the source ACS would check the communication with its own AER whether correct or not.

1. If it's wrong, source ACS would reply a DIAG_INFO-Response message in which its Data filed is filled with 2 for fault cannot be repaired and alarm to administrator to deal with this problem.

2. If it's right, source ACS would RENEW all the registration information, prefix information and state machine information to its all AERs. After that, source ACS will reply a DIAG_INFO-Response message in which its Data filed is filled with 1 for all runs correctly after repairing.

5. ACS-AER Communication Protocol

ACS would periodically deploy AD registration information, AD prefix information, and state machine information of relevant ADes to its all AERs to guarantee all information are latest. And ACS also would deploy the tag information to its all AERs periodically.

5.1. Deployment, Request and Response of AD Registration information

5.1.1. Deployment of AD Registration Information

After connecting with AER, ACS deploys the AD Registration Information (REG_INFO-Deploy) to AER peroidically. I Type is REG_INFO. S Type is Announcement. Operation is NULL when some ADes' information are joined, left or updated and Operation is RENEW when all ADes' information are deployed. Acknowledgement is 0. Data field is one or more ARI_Rec.

It should be noted that when there are two ARI_Recs in Data fields responding to the same AD, one may effect right now and the other effects after passing Effecting Time. When AER receives this message, all of them should be restored in the trust alliance list and AER MUST process them orderly. Since the protocol processes the records in sequence, it is required that the ARI_Rec effecting at the current time for the same member AD should appear in front of another updating ARI_Rec.

When receiving a non-RENEW packet, if it cannot resolve this message, AER could send a REG_INFO-Request message to acquire the latest AD registration information.

When AER can resolve this message correctly, it SHOULD:

1. Compare the Transaction Number in this packet with Transaction Number received from the same ACS before. If bigger, AER WOULD accept every ARI_Rec and process them as follows. Otherwise, AER would discard this packet and send a REG_INFO-RequestAll message to acquire the lastest information of AD registration information.

2. Process every ARI_Rec: - If Action is ADD and the record does not exist in its maintained trust alliance list, AER would add this record to its trust alliance list. - If Action is ADD and the record exists in its maintained trust alliance list but ACS Address is changed, AER would add this record to its trust alliance list and delete original record after passing Effecting Time in this ARI_Rec. - If Action is ADD and the record exists in its maintained trust alliance list and ACS Address is not changed, AER would do nothing. - If Action is DEL and the record exists in its maintained trust alliance list, AER would remove this record from its trust alliance list after passing Effecting Time in this ARI_Rec.

3. If a change is made in step 2, the update should take effect after passing the Effecting Time, which acts on the data plane. If the Effecting Time is earlier than the current time or is all 0, it will take effect immediately.

AER acts as following when receiving a RENEW packet. When ACS initiates RENEW, it would send a RENEW meesge with which the first bit of Operation field is 1. The second bit of Operation field identifies the begin of a procedure of RENEW and the third bit of Operation field identifies the end of a procedure of RENEW. ACS MUST NOT send a RENEW packet with which the first bit of Operation field is 0 in RENEWing. AER MUST process this procedure of RENEW after received all RENEW packets.

When AER can resolve this packet correctly, it SHOULD:

1. Compare the Transaction Number in this packet with Transaction Number received from the same ACS before. If bigger, AER would accept every ARI_Rec and process them as follows. Otherwise, AER would discard this packet and send a REG_INFO-RequestAll message to acquire the lastest information of AD registration information.

2. Process every ARI_Rec: - If the record does not exist in its maintained trust alliance list, AER would add this record to its trust alliance list. - If the record exists in its maintained trust alliance list but ACS Address is changed, AER

would add this record to its trust alliance list and delete original record after passing Effecting Time in this ARI_Rec. - If the record exists in its maintained trust alliance list and ACS Address is not changed, AER would do nothing. - If there are some records in the original trust alliance list that do not appear in the Data field during this RENEW process, they will be deleted immediately.

3. If a change is made in step 2, the update should take effect after passing the Effecting Time, which acts on the data plane. If the Effecting Time is earlier than the current time or is all 0, it will take effect immediately.

5.1.2. Request of AD Registration Information

The request is sent by AER to ACS. There are two types of request of AD Registration Information message. When querying the information of all member ADs of the trust alliance, the type is REG_INFO-RequestAll and REG_INFO-Request is used when querying the information of partial member ADs of the trust alliance.

Field	**Value**
Version	1
Alliance	The sub-trust alliance number.
I Type	REG_INFO
S Type	REQUEST: for querying partial member ADs and S Type is REQUEST_ALL: for querying all member ADs.
Operation	NULL
Total Length	The length of this message.
Number of Records	S Type = REQUEST: the number of ADID_Recs in Data field. S Type = REQUEST_ALL: 0.
Transaction Number	The last Transaction Number add 1. AER would maintain a global Transaction Number for packets sent out to ACS where I Type is REG_INFO and AER would keep it increasing monotonic.
Acknowledgement Number	0
Data	S Type = REQUEST: one or more ADID_Recs. S Type = REQUEST_ALL: None.

Table 7

When processing REG_INFO-Request(ALL) message, ACS would reply REG_INFO-NAK to AER if it holds some fields are wrong. For example, AER requests one ARI_Rec that does not exist. Otherwise, REG_INFO-ACK message will be replyed. ACS WOULD process as follows:

1. ACS SHOULD compare the Transaction Number in this packet with Transaction Number received from the same AER before. If

bigger, ACS would process as step 2. Otherwise, AER WOULD discard this packet and send a REG_INFO-NAK message to AER.

2. ACS processes every ADID_Rec. If the AD exists in its maintained trust alliance list, ACS would mark this record as "Reply". Otherwise ACS would mark this record as "Negative Reply". Especially, all records would be marked with "Reply" when Operation field is REQUEST_ALL.

3. If any case in step 2 is marked with "Negative Reply", ACS would construct a REG_INFO-NAK message to reply to the AER. Otherwise, a REG_INFO-ACK message is constructed to reply the AD registration information of all members marked with "Reply" to the AER.

5.1.3. Reponse of AD Registration Information

AD registration information response includs two types. That is REG_INFO-ACK and REG_INFO-NAK. ACS will reply to AER according to the request of registration information sent by AER to ACS.

Field	Value
Version	1
Alliance	The sub-trust alliance number.
I Type	REG_INFO
S Type	ACK: representing affirmative acknowledgement. NAK: representing negative acknowledgement.
Operation	NULL: REG_INFO-Request message. RENEW: REG_INFO-RequestAll.
Total Length	The length of this message.
Number of Records	S Type = ACK: the number of ARI_Recs in Data field. S Type = REQUEST_ALL: 0.
Transaction Number	The last Transaction Number add 1. ACS would maintain a global Transaction Number for packets sent out to AER where I Type is REG_INFO and ACS would keep it increasing monotonic.
Acknowledgement Number	The Transaction Number of the response corresponding request.
Data	S Type = ACK: one or more ARI_Recs. S Type = NAK: a 32-bit error code defined at Section 3. There is no boundary identification between these ARI_Recs, which requires that the implementation of the protocol can process each record sequentially until the end of this message.

Table 8

It should be noted that when there are two ARI_Recs in Data fields responding to the same AD, one may effect right now and the other effects after passing Effecting Time. When AER receives this message, all of them should be restored in the trust alliance list and AER MUST process them orderly. Since the protocol processes the records in sequence, it is required that the ARI_Rec effecting at the current time for the same member AD should appear in front of another updating ARI_Rec.

When receiving a non-RENEW REG_INFO-ACK message, if it holds that some fields are wrong, AER could send a REG_INFO-RequestAll message to acquire the latest AD registration information. Otherwise, AER would act as follows.

1. AER SHOULD compare the Transaction Number in this packet with Transaction Number received from the same ACS before. If bigger, AER would process them as follows. Otherwise, AER would discard this packet and send a REG_INFO-RequestAll message to acquire the lastest information of AD registration information.

2. AER WOULD processes every ARI_Rec: - If Action is ADD and the record does not exist in its maintained trust alliance list, AER would add this record to its trust alliance list. - If Action is ADD and the record exists in its maintained trust alliance list but ACS Address is changed, AER would add this record to its trust alliance list and delete original record after passing Effecting Time in this ARI_Rec. - If Action is ADD and the record exists in its maintained trust alliance list and ACS Address is not changed, AER would do nothing. - If Action is DEL and the record exists in its maintained trust alliance list, AER would remove this record from its trust alliance list after passing Effecting Time in this ARI_Rec.

3. If a change is made in step 2, the update should take effect after passing the Effecting Time, which acts on the data plane. If the Effecting Time is earlier than the current time or is all 0, it will take effect immediately.

AER acts as following when receiving a RENEW REG_INFO-ACK message. When ACS initiates RENEW, it would send a RENEW meesge with which the first bit of Operation field is 1. The second bit of Operation field identifies the begin of a procedure of RENEW and the third bit of Operation field identifies the end of a procedure of RENEW. ACS MUST NOT send a RENEW packet with which the first bit of Operation field is 0 in RENEWing. AER MUST process this procedure of RENEW after received all RENEW packets.

When AER can resolve this packet correctly, it SHOULD:

1. Compare the Transaction Number in this packet with Transaction Number received from the same ACS before. If bigger, AER would accept every ARI_Rec and process them as step 2. Otherwise, AER

would discard this packet and send a REG_INFO-RequestAll message to acquire the lastest information of AD registration information.

2. Process every ARI_Rec: - If the record does not exist in its maintained trust alliance list, AER would add this record to its trust alliance list. - If the record exists in its maintained trust alliance list but ACS Address is changed, AER would add this record to its trust alliance list and delete original record after passing Effecting Time in this ARI_Rec. - If the record exists in its maintained trust alliance list and ACS Address is not changed, AER would do nothing. -If there are some records in the original trust alliance list that do not appear in the Data field during this RENEW process, they will be deleted immediately.

3. If a change is made in step 2, the update should take effect after passing the Effecting Time, which acts on the data plane. If the Effecting Time is earlier than the current time or is all 0, it will take effect immediately.

When AER receives an REG_INFO-NAK message, it could send a REG_INFO-RequestAll message to ACS to acquire the latest AD registration information.

5.2. Deployment, Request and Reply of AD Prefix Information

5.2.1. Deployment of AD Prefix Information

AD prefix information deployment (PFX_INFO-Deploy) is sent from ACS to AER. ACS fills in the following values for each field:

Field	**Value**
Version	1
Alliance	The sub-trust alliance number.
I Type	AD_PREFIX_INFO
S Type	DEPLOYMENT
Operation	NULL: to publish partial update information of member ADs' prefix. RENEW: to publish all member ADs' prefix.
Total Length	The length of this message.
Number of Records	The number of API_Recs in Data field.
Transaction Number	The last Transaction Number add 1. ACS would maintain a global Transaction Number for packets sent out to AER where I Type is AD_PREFIX_INFO and ACS would keep it increasing monotonic.
Acknowledgement Number	0

Field	Value
Data	One or more API_Recs. There is no boundary identification between these API_Recs, which requires that the implementation of the protocol can process each record sequentially until the end of this message.

Table 9

It should be noted that when there are two ARI_Recs in Data fields responding to the same AD, one may effect right now and the other is update message for ADD or DEL effecting after the Effecting Time. For example, if the current time is 5 and there are two records corresponding to the prefix P, in which the Effecting Time of record R1 is 1, the action is ADD, the Effecting Time of record R2 is 7 and the action is DEL, then it indicates that the prefix P is currently valid effective from time 1 and becomes invalid at time 7. When ACS or AER receives this message, all of them should be restored in the database and ACS should send them all when deploying. Since the protocol processes the records in sequence, it is required that the API_Rec effecting at the current time for the same member AD should appear in front of another updating API_Rec.

When receiving a non-RENEW PFX_INFO-Deploy message, if it holds that some fields are wrong, for example, it requires to delete a API_Rec that does not exist or to add some prefix that is conflict with other member ADs, AER could send a request message to acquire the latest AD prefix information. Otherwise, AER would act as follows.

1. AER SHOULD compare the Transaction Number in this packet with Transaction Number received from the same ACS before. If bigger, AER WOULD process them as step 2. Otherwise, AER would discard this packet and send a PFX_INFO-RequestAll message to acquire the lastest information of AD prefix information.

2. AER processes every API_Rec: - If Action is ADD and the record does not exist in its maintained prefix list, AER would add this record to its prefix list. - If Action is ADD and the record exists in its maintained prefix list, AER would do nothing. - If Action is DEL and the record exists in its maintained prefix list, AER would remove this record from its prefix list after Effecting Time.

3. If a change is made in step 2, the update should take effect after the Effecting Time, which acts on the data plane. If the Effecting Time is earlier than the current time or is all 0, it will take effect immediately.

AER acts as following when receiving a RENEW PFX_INFO-Deploy message. When ACS initiates RENEW, it would send a RENEW meesge with which the first bit of Operation field is 1. The second bit of Operation field identifies the begin of a procedure of RENEW and the third bit of Operation field identifies the end of a procedure of RENEW. ACS MUST NOT send a RENEW packet with which the first bit of Operation field is 0 in RENEWing. AER SHOULD uniformly process all packets in this RENEW process after receiving all RENEW packets.

1. AER SHOULD compare the Transaction Number in this packet with Transaction Number received from the same ACS before. If bigger, AER WOULD process as step 2. Otherwise, AER would discard this message and send a PFX_INFO-RequestAll message to acquire the lastest information of AD prefix information.

2. AER processes every API_Rec: - If the record does not exist in its maintained prefix list, AER would add this record to its trust alliance list. - If the record exists in its maintained prefix list, AER would do nothing. - If there are some records in the original prefix list that do not appear in the Data field during this RENEW process, these records will be deleted immediately.

3. If a change is made in step 2, the update should take effect after passing the Effecting Time, which acts on the data plane. If the Effecting Time is earlier than the current time or is all 0, it will take effect immediately.

5.2.2. Request of AD Prefix Information

AD prefix information request (PFX_INFO-RequestAll) is sent from AER to ACS to query some member ADs', including itself, all latest AD prefix information.

AER fills in the following values for each field:

Field	Value
Version	1
Alliance	The sub-trust alliance number.
I Type	AD_PREFIX_INFO
S Type	REQUEST_ALL: querying from ACS the latest AD prefix information of all member ADs.
Operation	NULL
Total Length	The length of this message.
Number of Records	0
Transaction Number	The last Transaction Number add 1. AER would maintain a global Transaction Number for packets sent out to ACS where I Type is AD_PREFIX_INFO and AER would keep it increasing monotonic.
	0

Field	Value
Acknowledgement Number	
Data	None

Table 10

When receiving a PFX_INFO-RequestAll message, if it holds that some fields are wrong, ACS could send a PFX_INFO-NAK. Otherwise, ACS would act as follows. The specific construction methods of PFX_INFO-ACK and PFX_INFO-NAK are described in Section 5.2.3.

1. ACS SHOULD compare the Transaction Number in this packet with Transaction Number whose I Type is PFX_INFO received from the same AER before. If bigger, ACS WOULD process them as step 2. Otherwise, ACS would discard this packet and send a PFX_INFO-NAK message.

2. ACS processes every ADID_Rec. If AD exists in the maintained trust alliance list, ACS would mark this record as "Reply". Otherwise, ACS wourld mark this rocord as "Negative Reply". Particularly, all records are marked with "Reply" when S Type is REQUEST_ALL.

3. If any case in step 2 is marked with "Negative Reply", ACS would construct a PFX_INFO-NAK message to reply to the AER. Otherwise, a PFX_INFO-ACK message is constructed to reply the AD prefix information of all members marked with "Reply" to the AER.

5.2.3. Response of AD Prefix Information

AD prefix information response includs two types. That is PFX_INFO-ACK and PFX_INFO-NAK. According to the request sent by AER, if some fields are wrong, ACS will reply NAK, in which the error code is "parameter error". If a non-existent member AD is queried, the error code is "the requested member AD does not exist", which defined as before will not be repeated. The following mainly introduces PFX_INFO-ACK response. ACS fills in the following values for each field:

Field	Value
Version	1
Alliance	The sub-trust alliance number.
I Type	AD_PREFIX_INFO
S Type	ACK: representing affirmative acknowledgement. NAK: representing negative acknowledgement.
Operation	RENEW: replying the latest AD prefix information to AER.
Total Length	The length of this message.

Field	Value
Number of Records	S Type = ACK: the number of API_Rec in Data field. S Type = NAK: 0
Transaction Number	The last Transaction Number add 1. ACS would maintain a global Transaction Number for packets sent out to AER where I Type is AD_PREFIX_INFO and ACS would keep it increasing monotonic.
Acknowledgement Number	The Transaction Number of the response corresponding request.
Data	S Type = ACK: One or more latest requested API_Rec. S Type = NAK: a 32-bit error code defined in Section 3. There is no boundary identification between these API_Recs, which requires that the implementation of the protocol can process each record sequentially until the end of this message.

Table 11

When receiving a non-RENEW PFX_INFO-ACK message which is the positive reply to the request of AD prefix sent from ACS to AER, if it holds that some fields are wrong, AER could send a request message to acquire the latest AD prefix information. Otherwise, AER would act as follows.

1. AER SHOULD compare the Transaction Number in this packet with Transaction Number whose I Type is PFX_INFO received from the same ACS before. If bigger, AER would process them as follows. Otherwise, AER would discard this packet and send REG_INFO-RequestAll and PFX_INFO-RequestAll messages to acquire the lastest information.

2. AER processes every API_Rec: - If Action is ADD and the record does not exist in its maintained prefix list, AER would add this record to its prefix list. - If Action is ADD and the record exists in its maintained prefix list, AER would do nothing. - If Action is DEL and the record exists in its maintained prefix list, AER would remove this record from its prefix list after Effecting Time.

3. If a change is made in step 2, the update should take effect after the Effecting Time, which acts on the data plane. If the Effecting Time is earlier than the current time or is all 0, it will take effect immediately.

AER acts as following when receiving a RENEW PFX_INFO-ACK message. When ACS initiates RENEW process, it would send a RENEW meesge with which the first bit of Operation field is 1. The second bit of Operation field identifies the begin of a procedure of RENEW and the third bit of Operation field identifies the end of a procedure of RENEW. ACS MUST NOT send a RENEW packet with which the first bit of

Operation field is 0 in RENEW process. AER SHOULD uniformly process all packets in this RENEW process after receiving all RENEW packets.

1. AER SHOULD compare the Transaction Number in this packet with Transaction Number whose I Type is PFX_INFO received from the same ACS before. If bigger, AER WOULD process as step 2. Otherwise, AER would discard this message and send REG_INFO-RequestAll and PFX_INFO-RequestAll messages to acquire the lastest information.

2. AER processes every API_Rec. All Action in API_Recs is ADD during RENEW process. - If the record does not exist in its maintained prefix list, AER would add this record to its trust alliance list. - If the record exists in its maintained prefix list, AER would do nothing. - If there are some records in the original prefix list that do not appear in the Data field during this RENEW process, these records will be deleted immediately.

3. If a change is made in step 2, the update message should take effect after the Effecting Time, which acts on the data plane. If the Effecting Time is earlier than current time or is all 0, it will take effect immediately.

When AER receives an PFX_INFO-NAK message, it could send REG_INFO-RequestAll and PFX_INFO-RequestAll messages to ACS to acquire the latest AD registration information and AD prefix information.

5.3. Deployment, Request and Response of State Machine Information

5.3.1. Deployment of State Machine Information

State machine information deployment (SM_INFO-Deploy) is sent from ACS to AER. ACS fills in the following values for each field:

Field	**Value**
Version	1
Alliance	The sub-trust alliance number.
I Type	SM_INFO
S Type	DEPLOYMENT
Operation	NULL: to publish the partial update of state machine maintained by the pair of this AD and another AD and Operation is RENEW: to publish wholesome update of state machine maintained by the pair of this AD and another AD.
Total Length	The length of this message.
Number of Records	The number of SMI_Recs in Data field

Field	**Value**
Transaction Number	The last Transaction Number add 1. ACS would maintain a global Transaction Number for packets sent out to AER where I Type is SM_INFO and ACS would keep it increasing monotonic.
Acknowledgement Number	0
Data	One or more SMI_Recs. There is no boundary identification between these ARI_Recs, which requires that the implementation of the protocol can process each record sequentially until the end of this message.

Table 12

It should be noted that state machine is responding to a ordered AD pair. The state machine information mastered by ACS includes the state machine information from this AD to another member AD, and the state machine information from another member AD to this AD. When ACS deployment is partially updated, only some changed or newly added state machines are deployed. When ACS deploys the update of RENEW message, it is necessary to deploy all existing and updated information. For the same ordered AD pair, there cannot be two or more SMI_Recs using the same SM_ID in Data field. In addition, there are two actions for SMI_Rec: one is to add a SM whose SM_ID is bigger than current using state machine. The second is to modify an existing state machine whose SM_ID equals to current using state machine. Both of them are using Action ADD. Here we requires only Transition Interval and Expiring Time can be updated.

When receiving a non-RENEW SM_INFO-Deploy message sent from ACS to AER, if it holds that some fields are wrong, for example, Action is DEL or SM_ID is smaller than current state machine in using, AER could send a request message to acquire the latest information. Otherwise, AER would act as follows.

1. AER SHOULD compare the Transaction Number in this packet with Transaction Number whose I Type is SM_INFO received from the same ACS before. If bigger, AER WOULD process them as step 2. Otherwise, AER would discard this packet and send REG_INFO-RequestAll and request messages to acquire the lastest information.

2. AER processes every SMI_Rec: - If SM_ID equals to the current using state machine, AER should update the state machine in use. - If SM_ID bigger than the current using state machine, AER should add this state machine to its list.

3. If a change is made in step 2, the update message should take effect after the Effecting Time, which acts on the data plane.

If the Effecting Time is earlier than the current time or is all 0, it will take effect immediately.

AER acts as following when receiving a RENEW SM_INFO-Deploy message. When ACS initiates RENEW process, it would send a RENEW meesge with which the first bit of Operation field is 1. The second bit of Operation field identifies the begin of a procedure of RENEW and the third bit of Operation field identifies the end of a procedure of RENEW. ACS MUST NOT send a RENEW packet with which the first bit of Operation field is 0 in RENEW process. AER SHOULD uniformly process all packets in this RENEW process after receiving all RENEW packets.

1. AER SHOULD compare the Transaction Number in this packet with Transaction Number whose I Type is SM_INFO received from the same ACS before. If bigger, AER WOULD process as step 2. Otherwise, AER would discard this message and send a request messages to acquire the lastest information.

2. AER processes every SMI_Rec. - If SM_ID equals to the current using state machine, AER should update the state machine in use. - If SM_ID bigger than the current using state machine, AER should add this state machine to its list. - If there are some records of state machines in use that do not appear in the Data field during this RENEW process, these state machines will be deleted immediately.

3. If a change is made in step 2, the update message should take effect after the Effecting Time, which acts on the data plane. If the Effecting Time is earlier than current time or is all 0, it will take effect immediately.

5.3.2. Request of State Machine Information

State machine information request (SM_INFO-Request) is sent from AER to ACS. AER fills in the following values for each field:

Field	**Value**
Version	1
Alliance	The sub-trust alliance number.
I Type	SM_INFO
S Type	REQUEST: querying the state machines maintained by the pair of this AD to another member AD and vice versa. These member ADs are specified by ADID_Rec defined in Data field. REQUEST_ALL: querying all state machines maintained by this AD with other member ADs.
Operation	NULL
Total Length	The length of this message.

Field	**Value**
Number of Records	S Type = REQUEST: the number of ADID_Rec in Data field. S Type = REQUEST_ALL: 0.
Transaction Number	The last Transaction Number add 1. AER would maintain a global Transaction Number for packets sent out to ACS where I Type is SM_INFO and AER would keep it increasing monotonic.
Acknowledgement Number	0
Data	S Type = REQUEST: One or more ADID_Recs. S Type = REQUEST_ALL: none. There is no boundary identification between these ADID_Recs, which requires that the implementation of the protocol can process each record sequentially until the end of this message.

Table 13

For example, let this AD is AD1. When any ADID_Rec including in Data field, defined as AD2, it means that AER will request the SM(AD1, AD2) and SM(AD2, AD1). When ACS replies, it will reply these two state machines both.

When receiving a SM_INFO-Request(All) message, if it holds that some fields are wrong, ACS could send a PFX_INFO-NAK. Otherwise, ACS would act as follows. The specific construction methods of SM_INFO-ACK and SM_INFO-NAK are described in secion 3.2.3.3.

1. ACS SHOULD compare the Transaction Number in this packet with Transaction Number whose I Type is SM_INFO received from the same AER before. If bigger, ACS WOULD process them as step 2. Otherwise, ACS would discard this packet and send a SM_INFO-NAK message.

2. ACS processes every ADID_Rec. If AD exists in the maintained trust alliance list, ACS would mark this record as "Reply". Otherwise, ACS wourld mark this rocord as "Negative Reply". Particularly, all records are marked with "Reply" when S Type is REQUEST_ALL.

3. If any case in step 2 is marked with "Negative Reply", ACS would construct a SM_INFO-NAK message to reply to the AER. Otherwise, a SM_INFO-ACK message is constructed to reply the state machine information of all members marked with "Reply" to the AER.

5.3.3. Response of State Machine Information

State machine information response includs two types. That is SM_INFO-ACK and SM_INFO-NAK. Both of them are sent from ACS to AER. ACS fills in the following values for each field:

Field	Value
Version	1
Alliance	The sub-trust alliance number.
I Type	SM_INFO
S Type	ACK: representing affirmative acknowledgement. NAK: representing negative acknowledgement.
Operation	RENEW: replying the latest state machine information to AER.
Total Length	The length of this message.
Number of Records	S Type = ACK: the number of SMI_Recs in Data field. S Type = NAK: 0.
Transaction Number	The last Transaction Number add 1. ACS would maintain a global Transaction Number for packets sent to AER where I Type is SM_INFO and would keep it increasing monotonic.
Acknowledgement Number	The Transaction Number of the response corresponding request.
Data	S Type = ACK: one or more latest requested SMI_Rec. S Type = NAK: a 32-bit error code defined in Section 3. There is no boundary identification between these ADID_Recs, which requires that the implementation of the protocol can process each record sequentially until the end of this message.

Table 14

When receiving a non-RENEW SM_INFO-ACK message which is the positive reply to the request of AD prefix sent from ACS to AER, if it holds that some fields are wrong, AER could send a request message to acquire the latest state machine information. Otherwise, AER would act as follows. 1. AER SHOULD compare the Transaction Number in this packet with Transaction Number whose I Type is PFX_INFO received from the same ACS before. If bigger, AER WOULD process them as step 2. Otherwise, AER would discard this packet and send a SM_INFO-RequestAll message to acquire the lastest information. 2. AER processes every SMI_Rec: - If SM_ID equals to the current using state machine, AER should update the state machine in use. - If SM_ID bigger than the current using state machine, AER should add this state machine to its list. 3. If a change is made in step 2, the update should take effect after the Effecting Time, which acts on the data plane. If the Effecting Time is earlier than the current time or is all 0, it will take effect immediately.

AER acts as following when receiving a RENEW SM_INFO-ACK message. When ACS initiates RENEW process, it would send a RENEW meesge with which the first bit of Operation field is 1. The second bit of Operation field identifies the begin of a procedure of RENEW and the third bit of Operation field identifies the end of a procedure of RENEW. ACS MUST NOT send a RENEW packet with which the first bit of Operation field is 0 in RENEW process. AER SHOULD uniformly process all packets in this RENEW process after receiving all RENEW packets.

1. AER SHOULD compare the Transaction Number in this packet with Transaction Number whose I Type is SM_INFO received from the same ACS before. If bigger, AER WOULD process as step 2. Otherwise, AER would discard this message and send a SM_INFO-RequestAll message to acquire the lastest information.

2. AER processes every API_Rec. All Action in API_Recs is ADD during RENEW process. - If SM_ID equals to the current using state machine, AER should update the state machine in use. - If SM_ID bigger than the current using state machine, AER should add this state machine to its list. - If there are some records of state machines in use that do not appear in the Data field during this RENEW process, these state machines will be deleted immediately.

3. If a change is made in step 2, the update message should take effect after the Effecting Time, which acts on the data plane. If the Effecting Time is earlier than current time or is all 0, it will take effect immediately.

When AER receives an SM_INFO-NAK message, it could send a SM_INFO-RequestAll message to ACS to acquire the latest state machine information.

5.4. Request and Response of Keep-alive Information

In SAVA-X, ACS will periodically send Keep-alive request to query the availability of AER in SAVA-X mechanism.

5.4.1. Request of Keep-alive Information

Keep-alive information request (ALIVE_INFO-Request) is sent by ACS to test the viability of AER. AER would reply to ACS when receiving a ALIVE_INFO-Request message. ACS considers that AER has gone wrong if it does not receive a response from AER within 60 seconds and ACS notifies the AD administrator of the failure information by email. ACS would keep sending ALIVE_INFO-Request to the fault AER at the same time. The filling values of each field in ACS request are as follows:

Field	Value
Version	1
Alliance	The sub-trust alliance number.
I Type	ALIVE_INFO
S Type	REQUEST
Operation	NULL
Total Length	The length of this message.
Number of Records	0
Transaction Number	The last Transaction Number add 1. ACS would maintain a global Transaction Number for packets sent to AER where I Type is ALIVE_INFO and would keep it increasing monotonic.
Acknowledgement Number	0
Data	None

Table 15

ACS considers that AER has gone wrong if it does not receive a response from AER within 60 seconds and ACS notifies the AD administrator of the failure information by email. ACS would consider that AER has recovered from failure when AER reply to the request correctly. ACS performs the following steps to update AER:

1. Keep time synchronization between AER and ACS.

2. Deploy AD registration information, AD prefix information and state machine information to AER by the way of RENEW message.

5.4.2. Response of Keep-alive Information

Keep-alive information response (ALIVE_INFO-Response) is sent by AER to reply the ALIVE_INFO-Request message.

In response to ALIVE_INFO-Request, AER fills in the following values for each field in the response:

Field	Value
Version	1
Alliance	The sub-trust alliance number.
I Type	ALIVE_INFO
S Type	ACK
Operation	NULL
Total Length	The length of this message.
Number of Records	0
Transaction Number	The last Transaction Number add 1. AER would maintain a global Transaction Number for packets

Field	Value
	sent to ACS where I Type is ALIVE_INFO and would keep it increasing monotonic.
Acknowledgement Number	0
Data	None

Table 16

6. Deployment of Tag Information

Tag information deployment (TAG_INFO-Deploy) is sent from ACS to AER and AER would add, verify and remove the tag to packet. When using sub trust alliance level tags and AD_V tags, the primary address domain ACS needs to distribute these two tags to the ACS of the boundary address domain first, and then the boundary address domain ACS will distribute these tags to their respective address domains' AERs. The sub trust alliance tag is used in the data plane to cross different address domain levels. The AD_V tag is used in the data plane when it is sent from the current address domain to the boundary address domain. Standard TAG_INFO is used in the data plane at the same level and under the same direct parent address field. The three types of tags use the same message format as follows.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|     Action    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source ADID_Rec                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Destination ADID_Rec                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Tag Len   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                              TAG                              ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Transition Interval                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Action: 8-bit filed. 1 for add (ADD=1) and 2 for delete (DEL=2).

Source ADID_Rec: Variable-length field. Refer to ADID_Rec in [SAVA-X-Control].

Destination ADID_Rec: Variable-length field. Refer to ADID_Rec.

Tag Len: The length of TAG. The equation for calculation is (Tag Len + 1) * 8 bits. The length of TAG MUST be multiple times of 8 bits. The maximum length is 128 bits and the minimum length is 32 bits. So the minimum of Tag Len is 0011.

TAG: Variable-length field. The actual Tag or packet signature.

Transition Interval: 32-bit, the milliseconds of interval of state transition.

When ACS announce tag to ACS or AER, it fills in the following values for each field:

Field	Value
Version	1
Alliance	The sub-trust alliance number.
I Type	TAG_INFO, ALLI_TAG_INFO or AD_V_TAG_INFO
S Type	ANNOUNCEMENT
Operation	NULL
Total Length	The length of this message.
Number of Records	The number of TAG_Rec in Data field.
Transaction Number	ACS would maintain a global Transaction Number for packets sent to ACS or AER where I Type is TAG_INFO and would keep it increasing monotonic. Acknowledgement Number is 0.
Acknowledgement Number	0
Data	One or more TAG_Recs. There is no boundary identification between these records, which requires that the implementation of the protocol can process each record sequentially until the end of this message.

Table 17

7. Security Consideration

This present memo doesnot find any security problem.

8. IANA Consideration

There are two tcp ports, 23160 and 23161, are used in implementing SAVA-X mechanism. Port 23160 is used for the communication between ACS and ACS. Port 23161 is used for the communication between ACS and AER.

9. Acknowledgements

Much of the content of this document is the expansion of the IETF [RFC5210] in inter-domain level. Thanks to the effort of pioneers.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5210] Wu, J., Bi, J., Li, X., Ren, G., Xu, K., and M. Williams, "A Source Address Validation Architecture (SAVA) Testbed and Deployment Experience", RFC 5210, DOI 10.17487/RFC5210, June 2008, <https://www.rfc-editor.org/info/rfc5210>.

[RFC8200] Deering, S., Hinden, R., and RFC Publisher, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

10.2. Informative References

[SAVA-X-Control] Computer Science, Computer Science, and Institute for Network Sciences and Cyberspace, "Control Plane of Inter-Domain Source Address Validation Architecture", 2021.

Authors' Addresses

Ke Xu
Computer Science, Tsinghua University
Qinghuayuan street, Haidian District
Beijing
100084
China

Email: xuke@tsinghua.edu.cn

Jianping Wu
Computer Science, Tsinghua University
Qinghuayuan street, Haidian District
Beijing
100084
China

Email: jianping@cernet.edu.cn

Xiaoliang Wang
Computer Science, Tsinghua University
Qinghuayuan street, Haidian District

Beijing
100084
China

Email: wangxiaoliang0623@foxmail.com

Yangfei Guo
Institute for Network Sciences and Cyberspace, Tsinghua University
Qinghuayuan street, Haidian District
Beijing
100084
China

Email: guoyangfei@zgclab.edu.cn